NISTIR 8090

Measuring and Representing the Performance of Manufacturing Assembly Robots

Michael Shneier
Elena Messina
Craig Schlenoff
Frederick Proctor
Thomas Kramer
Joseph Falco

This publication is available free of charge from:
http://dx.doi.org/10.6028/NIST.IR.8090

National Institute of Standards and Technology
U.S. Department of Commerce

NISTIR 8090

Measuring and Representing the Performance of Manufacturing Assembly Robots

Michael Shneier
Elena Messina
Craig Schlenoff
Frederick Proctor
Thomas Kramer
Joseph Falco
Intelligent Systems Division
Engineering Laboratory

This publication is available free of charge from:
http://dx.doi.org/10.6028/NIST.IR.8090

November 2015

U.S. Department of Commerce
Penny Pritzker, Secretary

National Institute of Standards and Technology
Willie May, Under Secretary of Commerce for Standards and Technology and Director

Abstract

With the growth of robotic technology, there is a need for performance measures to characterize and compare robots and to help determine which features are most suited to a particular application. Robot systems are complex and involve a wide range of features and performance characteristics whose importance differs depending on the application domain. This paper describes a set of assembly performance measures associated with the different features along with an exploration of how one could represent this information. It organizes the assembly measures in reference to a taxonomy of assembly skills and tasks. Arranging the performance measures in this way will simplify the task of selecting a particular robot system for an assembly task by helping focus on those aspects of the task that are most critical.

Keywords: Manufacturing System; Performance Evaluation; Performance Measures; Robotics; Taxonomy

1. Introduction

According to the International Federation of Robotics [1], there is a strong and increasing demand for industrial robots. The market is expected to expand from the traditional automotive and heavy industry sectors to include general manufacturing, and the purchasers are expected to include not only large organizations but also small and medium-sized manufacturers. As the use of robots increases and as the people who are using them include more non-experts, there will be a parallel increase in the need for well-defined ways of selecting which robotic components will be most suitable for a given application domain. Robotic systems are typically complex, including robots, their controllers, end-effectors, sensors, safety systems, and programming environments. These components are supplied by different vendors resulting in the practice of companies to employ expert integrators to design and install their robot systems and to ensure that all the parts work together.

The user community of robotic systems is expanding to include smaller organizations that do not have sufficient resources to employ integrators. This group needs a principled way of determining which components are best for their environment, how well the components will work together, and the level of effort that will be needed to build and program the complete system. Providing a set of performance measures for the individual components of a robot system will allow users to select and measure only those features that are important to them. Vendors of subcomponents can use the measures to characterize their equipment and advertise its features. Organizing the performance measures according to the different aspects of a typical robot system will make it easier for users to select the appropriate measures and clearer for the users to understand the interactions between those measures and the need to explore particular interfaces between components.

Defining the key performance parameters is a crucial first step in developing and organizing performance measures. The National Institute of Standards and Technology (NIST) has adopted a use-case or task-based approach for this. The idea is to characterize the performance of a technology with respect to a domain-specific operation rather than targeting design-specific tests or measures. This approach leaves the design and solution space unconstrained while providing data about how well a given solution meets the task requirements. Hierarchical task decomposition to derive performance requirements has been applied to many domains, including autonomous vehicles [2] and urban search and rescue robots [2]. The urban search and rescue robots work is the most fully-developed instance of this methodology, resulting in over 100 user-defined task-based performance requirements which are being used to guide development of standard test methods. The methodology is being applied to other categories of response robots, including bomb-disposal and military applications. A similar approach – SCORE (System, Component, and Operationally-Relevant Evaluation) has been developed by NIST and used for measuring performance of emerging intelligent systems technologies (not necessarily robotics) [3].

Examples of new technologies to which SCORE has been successfully used include soldier-worn sensor systems and automated translation tools [4, 5].

This document adopts a taxonomy of assembly tasks to guide the development of metrics for evaluating the performance of robotic systems. Assembly is the area of focus since it is currently one of the manufacturing tasks that is least supported by robotics. The taxonomy builds upon a variety of prior work, melding semantic, geometric, and constraint-based approaches. The document also includes an exploration of how one could capture these metrics and measures in a formal knowledge representation. This report is organized as follows: Section 2 provides an overview of the assembly process and explains some of the reasons why it is difficult. Section 3 addresses knowledge representations that have been use in the assembly domain including taxonomies for assembly. Section 4 discusses performance metrics, while Section 5 relates them to the taxonomies. Section 6 explores the information requirements necessary to represent these robots "capabilities" and ways that this information has been represented in the literature. Sections 7 and 8 provide some discussion and conclusions.

2. Overview of Assembly

Assembly is an essential and complex step in the manufacturing process. It is one area where automation, especially robotics, has not seen wide adoption. At its most elemental, assembly consists of a series of operations that join together individual parts or subassemblies. However, due to tight tolerances, difficult orientation requirements or access, and the extensive use of a variety of tools and assistive devices to achieve the join operation, much of assembly has been beyond the abilities of current robotic systems. Put another way, "robots for assembly and machining are perhaps the most difficult case due to uncertainties in combination with force interaction." [6]

Boothroyd et al. [7] and others claim that assembly accounts for 30-50 % of all manufacturing costs, hence it is an important area that could benefit U. S. competitiveness if robotic systems were able to assist in more of the operations. According to Shi and Menassa [8], "While robotic automation has played a key role in the automotive industry and specifically in stamping, welding, material handling and painting over the last 30 years, currently there are no robotic assembly applications in the final assembly of a vehicle on a moving line in domestic automotive manufacturing plants." The International Federation of Robotics states that, although assembly applications account for about half of the production schedule in automotive manufacturing, it accounts for only 7.3 % of robot sales. [1]

Focusing specifically on the decomposition of tasks that comprise assembly operations, we will look at analyses and knowledge representations for assembly in this document. In his book "Mechanical assemblies: their design, manufacture, and role in product development" [9], Whitney takes an in-depth look at the design and manufacture of assemblies. His definition of assembly is constraint-oriented:

> An assembly is a chain of coordinate frames on parts designed to achieve certain dimensional relationships, called key characteristics, between some of the parts or between features on those parts.

Some key concepts that have applicability to our derivation of task-based performance metrics are summarized in the sections below.

Since we are interested in assembly from a performance metrics perspective, it is useful to include another quote from Whitney on the conditions for a successful assembly:

> The mechanics of part mating are governed by the geometry of the parts, the compliance of the parts and supports, the friction between parts as they move past each other during assembly, and the amount of lateral and angular error between the parts as the mating begins. The interplay of these factors determines whether assembly will be successful and how large will be the forces exerted on the parts by the tooling and each other.

2.1. Motion Aspects of Assembly

We specifically call out the motion aspects of assembly since they are a major component of the overall process and therefore are relevant to the performance requirements for robotic systems performing assembly operations. The range of spatial scales encountered in assembly operations and associated uncertainties are shown in Figure 1.

Two main motion types are involved during assembly operations. Gross motions are used to move the part over distances that are large compared to the part size. According to Whitney, about half of all assembly time is consumed by this type of motion. Gross motions are fast and do not typically require high accuracy, except as the part approaches its destination. The second type is fine motion, which is small compared to the size of a part and occurs when parts are touching during the mating stage itself. Errors in fine motions are typically too small to see, but can be detected with force sensors. As the allowed magnitude of the errors decreases, the cost of carefully positioning parts and designing special fixtures or features, such as chamfers, to reduce them rises dramatically. Types of fine motion errors are classified as

- Lateral Position
- Angular Orientation
- Jamming
- Wedging
- Screw thread mating: angular, threads being out of phase, and incorrect tightening
- Gear mating errors, both during side approach and spin-axis approach of mating

Whitney did not discuss a special case that requires fine manipulation prior to mating, a process that requires constrained gross motions. This could be required by having to insert a part to be

mated to another part that is difficult to access. Whitney does summarize different ways that a part may be brought to its final state of assembly, along with the degree of uncertainty in location relative to the part size. Understanding the magnitudes of position uncertainties for different categories of parts and assembly sub-tasks is useful in devising the performance metrics.

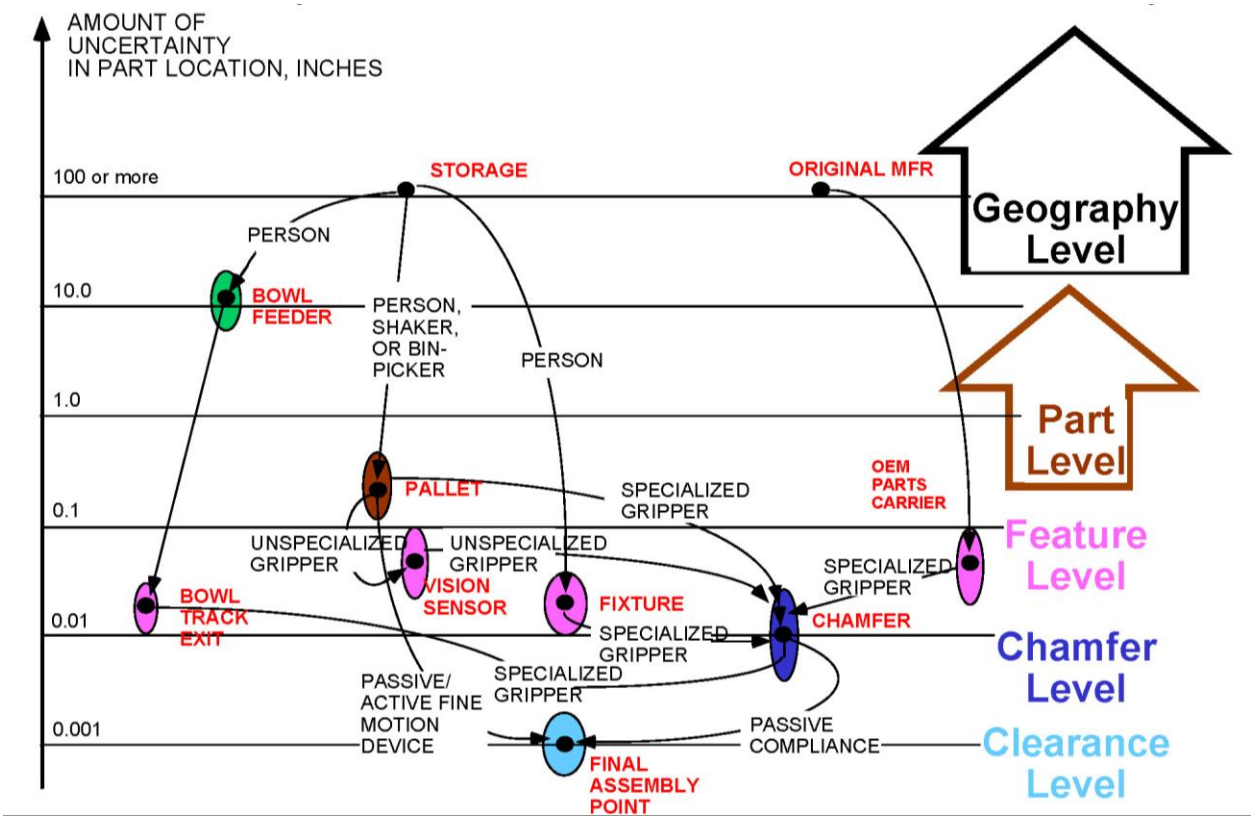

Figure 1: Location uncertainty relative to part size (Figure © Mechanical Assemblies: Their Design, Manufacture, and Role in Product Development by Whitney (2004, Figure 17-1). By permission of Oxford University Press, USA

For fine motion, an older census of assembly operations by Kondoleon [10] cited by Whitney, finds that peg-in-hole and screw operations account for over half of assembly joins. For this reason, peg-in-hole in particular (accounting for over 35 % of operations) has been closely analyzed.

A key dimension for this type of insertion is the **Clearance ratio,** c = (hole diameter – peg diameter)/ (hole diameter). According to a survey of dimensioning practice for rigid parts cited in Whitney [9], the clearance ratio varies only slightly (by a decade or less), and can be estimated by knowing the type of the part. This is an important consideration for setting performance expectations with respect to what tolerances robots may need to deal with in performing joining tasks.

Another important consideration is the fact that many general assembly operations use a moving line. Shi and Menassa [8] emphasize that ”new robotic technologies have to be advanced to enable the installation of parts on a moving assembly line.” This is an additional requirement for fully successful and robust robotic assembly solutions. For many applications, the robot systems' ability to track the parts on a moving line must be overlaid on the evaluation of the robot's performance of the individual assembly tasks. Some have addressed this challenge by researching and implementing mobile manipulators. [11]

2.2. Assembly Difficulty Measures

The Design for Assembly discipline was developed, starting in the 1970s, to address the large proportion of manufacturing cost attributable to the assembly process [7]. Design for Assembly (DFA) includes measures of assembly difficulty, as well as guidelines for product design that facilitate the assembly processes. Classification schemes from this discipline could be used to guide robotic assembly metrics. Indeed, DFA considerations include whether it will be possible to use humans, fixed automation, or flexible automation (robots). This area of study is relevant for our purposes, as the assembly operations or factors that create barriers to adopting robotics are enumerated.

The traditional view of how to allocate tasks to either mechanized or manual assembly is that it is determined by the volume and variety entailed by the task. High-volume, low variety is considered suitable for automation, whereas either low volume or high variety is still the province of manual assembly. See for example Figure 2 for a comparison of the relative cost profiles for manual versus fixed automation versus flexible automation by volume. In [9], the divide between low and high volume is set at around 100 000 units per year. The unit costs for different types of assembly methods are shown below. An additional consideration in the cost and flexibility equation, which thus far has not been discussed in depth in the literature, is material transport: hard automation, such as fixed conveyors, versus somewhat flexible approaches, such as automatic guided vehicles, or fully flexible mobile manipulators.

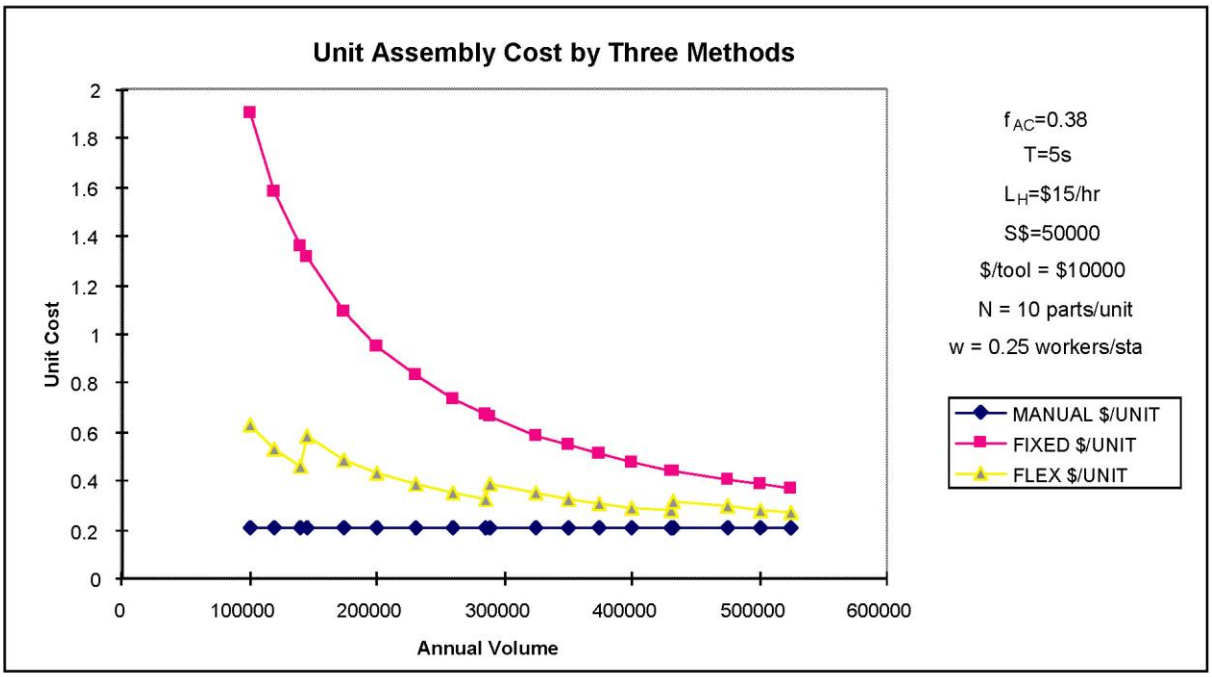

Figure 2: Comparison of assembly costs by different methods (Figure © Mechanical Assemblies: Their Design, Manufacture, and Role in Product Development by Whitney (2004, Figure 16-5). By permission of Oxford University Press, USA

Boothroyd found that certain factors contribute to the difficulty – and ultimately the cost – of performing an assembly operation. These include part feeding, orienting, handling, and inserting. Part attributes also contribute to the difficulty score. Others have also developed methods of scoring the difficulty of an assembly. The Hitachi Assembleability Evaluation Method [12] scored direction of insertion, number of non-insertion extra operations, and others in the cost function. Westinghouse developed an assembly difficulty calculator, which was modified by Ishii [13]. Others have focused on the costs of robotic assembly, taking into consideration challenges such as that robot manipulators typically lack a second arm or hand to hold a part during a join operation or have insufficient arm or gripper dexterity to orient a part, particularly if it is not accessed from the top down [14].

A brief summary of factors to be taken into account when calculating the difficulty of a given assembly task or the time required to complete it is presented in Table 1. For more details, see the above references.

Table 1: Factors influencing manual assembly difficulty or cost

Part handling difficulty	- Size
	- Thickness
	- Weight
	- Fragility
	- Flexibility
	- Slipperiness
	- Stickiness
	- Degree of symmetry, measured as $\alpha + \beta$, where
	  - α is the angle required to rotate the part about an axis normal to the insertion axis in order to return it to its starting configuration
	  - β is the same with respect to an axis about the insertion axis.

Acquisition difficulty based on shape	- Sphere - Cylinder - Hexagon - Cube - Rivet - Hex face peg - Square face peg - 180° peg (mirror symmetry: can be inserted 2 ways correctly) - 360° peg (non-symmetric; can only be inserted 1 way)
Necessity for using additional assistance or tools	- Two hands - Optical magnification - Mechanical assistance
Conditions that affect manual insertion time	- Part is secured immediately or after other operations - Insertion region is accessible - Insertion region is visible - Part can be easily aligned and positioned - A tool is needed - Part must be held after placement until other parts or fasteners are installed - Insertion operation is difficult

Fastener Type (in order of increasing difficulty)	- Washer - Pin - Retaining ring - Screw - Nut - Rivet
Fastening process time (in order of increasing difficulty for manual operation)	- Snap or press fit - Bending or crimping - Screwing - Polymer weld - Solder - Weld or braze - Adhesive
Insertion Direction (increasing difficulty/cost)	- Down from top - From the side - Angled or Twist - Up from below

There are many sources of errors within a robotic workcell that contribute to the cost or difficulty of performing the assembly actions and hence should be considered when developing performance metrics and test methods. These must either be dealt with through the robot's agility, including sensors and/or adaptive end-of-arm tooling or they must be controlled by external means. Whitney summarizes sources of robot assembly errors within a workcell as follows:

- Part Construction
- Part Jigging
- Jig Location
- Robot Accuracy and Calibration
- Tool Socket
- Part Grip
- Offline Model

3. Knowledge Representations for Assembly

We examine a range of knowledge representations that have been developed for assembly. These are useful for helping determine the decomposition of assembly into its constituent elements, which will then drive the taxonomy for the performance measures. The representations include semantic approaches, such as taxonomies, as well as geometric and constraint-based ones.

3.1. Taxonomies

There have been a number of attempts to develop taxonomies of robot operations. Some of these are reviewed below. Criteria for a good taxonomy for the purposes of this paper include coverage of all the robotic components, an indication of how they fit together, and the ability to adjust the granularity of the representation to focus on the critical aspects while still getting a measure of the

expected performance of the whole system. It is also desirable that the taxonomy be able to expand as the capabilities of robots change and that the taxonomy not incorporate preferred methods or kinds of equipment that may not be universally applicable.

A taxonomy developed by Seabra Lopes and Camarinha-Matos [15] is divided into operational resources (robots, grippers, fixtures, etc.), sensor resources (force and torque, presence, etc.), and resource storage units (e.g., tool changers). The resources manipulate artifacts which include parts, assembled objects, and unexpected items. The functionality of resources is described in terms of operators, such as approach, insert peg in hole, feed part, etc. The knowledge is encoded in a logical framework and reasoning and machine learning (programming by demonstration) are used to develop plans and to detect failures and re-plan.

In [16], Fiorentini et. al. describe an ontology for representing assemblies. This is more focused on the structure of mechanical assemblies and how to exchange this information between stakeholders than on the representation of the assembly operations. They use a combination of Web Ontology Language (OWL) and the Unified Modeling Language (UML). In addition to developing a semantic assembly information model, they incorporate reasoning capabilities.

In [17], Tenorth and Beetz describe KnowRob, which is a first-order knowledge representation based on description logics that provides specific mechanisms and tools for action-centered representations. Like the above, their knowledge is represented in OWL and uses the inherit classes, instances, and property constructs that are included in OWL. Their higher-level concepts are inspired by the Cyc ontology [18]. They have a general class called "ActionOnObject" which contains attributes such as objActed On and doneBy, and can be specialized into specific actions such as PutDown or PickUp.

In [19], Stipancic et al. present a context-aware system used within industrial environments. Of interest to this paper is the way that they represent their knowledge about assembly tasks. They use an ontology represented in OWL and leverage the work performed in the MASON [20] and OMTOMAS [21] projects. These projects attempted to build an ontology describing the field of production activities. Their ontology includes actions such as Checking, Feeding, Composing, Handling, Adjusting, and other Special Processes. They decompose these higher-level actions into sub-actions such as inspection, joining, and picking up. The authors extend these ontologies to include probabilistic information, basing their work on Bayesian Networks.

More recent work by Huckaby and Christensen [22] takes a similar approach to Seabra Lopes and Camarinha-Matos [15]. The authors define a taxonomy of tasks and skills as well as skill primitives, which are lower-level actions that are combined to achieve a skill. The taxonomy is more completely defined in their paper than in the Seabra Lopes work, although it is not clear that the earlier work did not include an equivalent set of tasks and skills. One thing that appears new is the use of constraints, such as required poses or durations in which a task must be accomplished. The taxonomy also explicitly includes primitives for coordination and sequencing of actions when

more than one robot is executing a task. The taxonomy can be extended or specialized as needed for a particular domain. It does not specify how actions should be executed, only the results that are expected. The taxonomy includes links to perception and control to enable verification that the task is executed properly. It does not appear to include error recovery. The taxonomy is illustrated in Figure 3.

Other work has taken a similar approach. For example, Pfrommer, et al. [23], break the production process into products, processes, and resources. Processes are broken further into tasks and skills, which are similar to those used by Huckaby and Christensen. Tasks have pre- and post-conditions as well as a duration. Skills are associated with machines that can execute parts of a process, such as being able to weld or transport parts. Processes include such actions as cutting, bending, movement, and fastening. Resources are machine tools, robots, conveyors, etc. Products are the intermediate or final results of the activity. Skills are the combination of an action and the machine that carries out the action so can, for example, include a robot's ability to move a part. Tasks are the steps needed to complete the activity.

Also focused on skills is the European Union funded project investigating "Skill-based Inspection and Assembly for Reconfigurable Automation Systems" (SIARAS) [24]. This project seeks to develop a complete representation of a production system's devices and associated skills to enable the automatic reconfiguration of the production system. To investigate skill representation, they begin with a taxonomy of tasks, the top levels of which are shown in Figure 4:

- *Skill*: A Skill represents an action that might be performed by a device as part of a production process. The SIARAS skill hierarchy is divided into six subcategories, the main one being called MainFunction. This subhierarchy is further broken out in Figure 4 and includes the definitions of manipulation, manufacturing, handling, and sensor functions.
- *Property*: Properties are aspects of devices and skills that the Skill Server can reason about. This subtree contains parameters for devices like sensors as well as physical parameters, communications interfaces, and quality criteria. As a result, the subtree is highly branched.
- *Physical Object*: Objects are the components of workcells. They include Devices, which are active and have skills, and Workpieces, which are passive and are manipulated by Devices. The device hierarchy is both deep and highly-branched to reflect the range of actual devices used in automation systems.

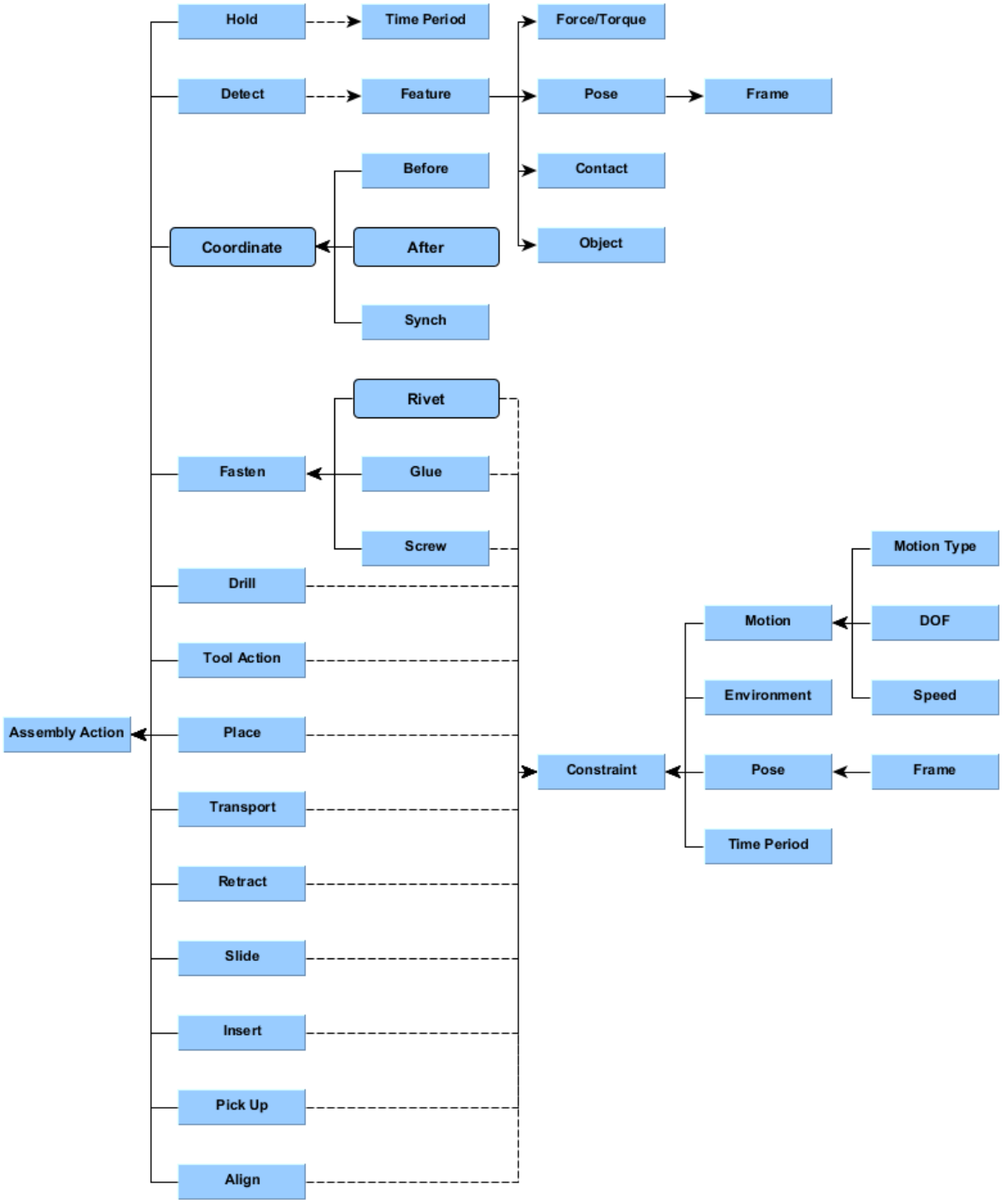

Figure 3: The taxonomy for the robot assembly domain defined by Huckaby and Christensen

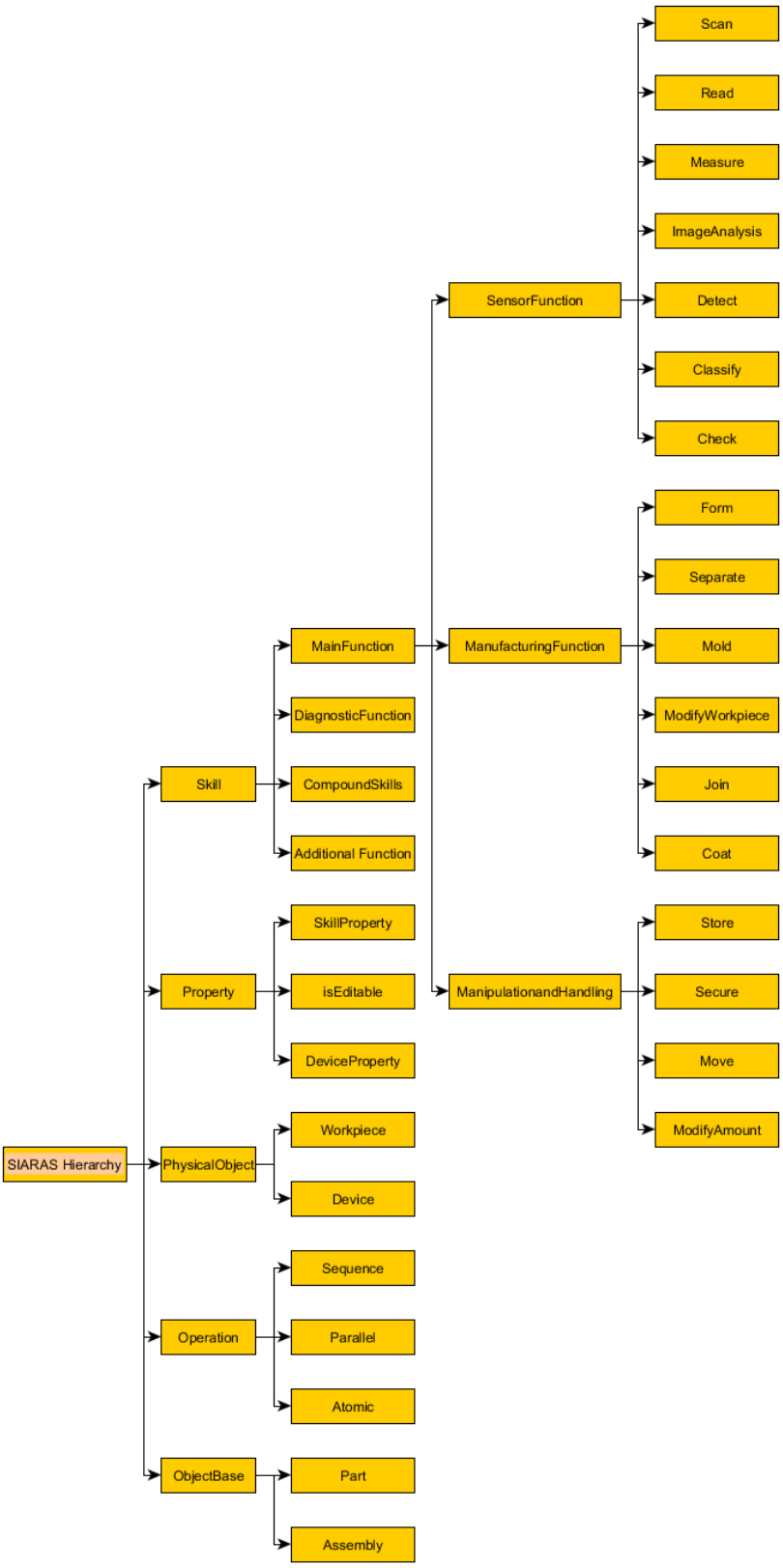

Figure 4. Part of the SIARAS hierarchy. In the full hierarchy, the nodes are further expanded

- *Operation*: Operations describe the tasks performed by a device. An operation can be the direct application of a single skill (called an Atomic operation), the invocation or two or more skills carried out at the same time (called a Parallel operation), or can involve two or more operations performed one after the other (called a Sequence operation).
- *Object Base*: The Object Base node provides a way to model Physical Objects. Each object is either a simple part or an assembly (a set of parts or other assemblies). It stores geometrical relations and dependencies of objects.

A task description is needed that describes the principal steps of the task and requirements, boundary conditions, or tolerances. Therefore, the SIARAS consortium developed a Flow Chart based approach, where the user can describe and parameterize the process description step by step.

Tallinen, Osuna, et al. [25] match a particular product's requirements for assembly processes to equipment process capabilities. In addition to the geometric and attribute information that is defined in other representations, they include information on handling, such as for feeding, picking, gripping, and positioning tasks. Information on insertion paths, positions, contact areas, and required tolerances are specified. They also include knowledge about precedence relations and tooling information.

Gerky and Matarić [26] define a taxonomy of multi-robot task allocation problems and use it to explore the complexity of possible solutions using methods from operations research, economics, scheduling, network flows, and combinatorial optimization. The taxonomy divides the kinds of problems using three axes: single-task robots vs. multi-task robots, single-robot tasks vs. multi-robot tasks, and instantaneous assignment vs. time-extended assignment. An analysis of existing algorithms that fit into the simpler of these classes shows that they are mostly equivalent within classes. The taxonomy can be used to select optimal or near-optimal algorithms for a given problem and architecture.

Drawing on a number of previous taxonomies for human-computer interaction and computer-supported cooperative work, Yanco and Drury [27] define a taxonomy for human-robot interaction. It is comprised of multiple axes in an attempt to cover the range of possibilities that can arise and is mostly focused on mobile robots. The factors included are: level of autonomy, ratio of number of people to number of robots, the amount and type of shared interaction, the kinds of decision support provided to the operators (humans), a measure of how critical completion of the task is, whether the robots used are homogeneous or heterogeneous, and a time-space (sub-) taxonomy. The time axis of the time-space taxonomy can be either synchronous or asynchronous, while the space axis can be either collocated or non-collocated. No specific measures are provided for quantitatively measuring where a particular system fits within the taxonomy.

Bloomfield, et al. [28] define a taxonomy of haptic actions for disassembly tasks. They explore the use of instrumented gloves to provide feedback to workers on representative tasks and use their results to develop the taxonomy. The classification has two major dimensions: the general type of

action performed and the type of force or torque required. Their taxonomy includes actions that discriminate amongst those requiring fine motor control, significant arm strength, tactile friction, cooperative two-handed tasks, braced two-handed tasks, manipulating a deformable object, tool-assisted tasks, and multiple finger tasks. The type of force required is divided into two types of force-only actions (e.g., pushing a button vs sanding a surface), two types of torque-only actions (e.g., turning a dial vs. using a wrench), and actions that require force and torque simultaneously (e.g., pulling and twisting to extract a piston from a cylinder.)

Tan, et al. [29] use hierarchical task analysis to decompose a task (electrical connector assembly) into subtasks. The decomposition uses primitive operations that are similar to the skills in the other taxonomies. They include retrieve, assemble, arrange, check, secure, hold, insert, temporarily fix, locate, release, place, etc. A human works in collaboration with robots to accomplish the tasks. A qualitative analysis is used to assign tasks to either the robot, the person, or to both collaboratively. The assignment is based on the perceived capabilities of each operator and the perceived requirements of the task. This may be followed by a quantitative procedure if the qualitative one does not provide a clear solution. They use productivity (assembly duration), quality (assembly error), human fatigue (human operator tiredness), and safety (human operation safety) as the criteria in the analysis. They also study the necessary motor and perception skills for the person for each step of the assembly and the safety aspects of the tasks. Human studies showed improvement in performance using a real human-robot system.

3.2. Other Representations for Assemblies

Beyond taxonomies, there is extensive literature defining different ways of representing assembly geometry and constraints, much of it to enable reasoning and automated plan generation. We review some representative approaches in this section. These representations capture feasible configurations of sub-assemblies and assemblies, hence can guide the further detailing of assembly operations as well as inform the performance metrics development process.

Homem de Mello and Sanderson [30] classify assembly representations as either explicit or implicit. Explicit ones establish a mapping from the assembly tasks into the elements of the representation. These include directed graphs and AND/OR graphs. Implicit representations are based on contact establishment conditions and on precedence relationships and consist of conditions that must be satisfied by the assembly sequences. They represent an assembly task as a directed graph of feasible assembly sequences comprised of a set of parts as nodes with edges representing the assembly actions that join nodes together. An assembly task is geometrically feasible if there is a collision-free path to bring the two subassemblies into contact. An assembly task is mechanically feasible if it is possible to establish the attachments that act on the contacts between the two subassemblies.

Lyons et al. [31] present an overview of representations and propose enhancements and extensions to the ISO 10303 standard for exchange of product model information (informally referred to as STEP) to support assembly modeling, analysis, and planning (Figure 5). The Lyons research is

not specifically aimed at robotic assembly, but their work addresses some of the key knowledge required to implement robotic assembly. They determine insertion difficulties by analyzing the mating constraints satisfied by the assembly of a component. Issues include access, motion trajectory, presence of locating features, and the characteristics of the joining process. In particular, their work proposes representation of assembly tolerances as attributes linked to the surface mating constraints. The basis for specifying acceptable tolerances is derived from the functionality of the joint. The Open Assembly Model [32] builds on Lyons et al. to provide a standard representation and exchange protocol for assembly and system-level tolerance information.

Balakirsky et al. [33] describe a knowledge representation for kitting, which is a subset of assembly. In kitting, a set of parts is placed into a container for delivery to the assembly workcell. Kits usually consist of all the parts needed for a subassembly, often presented in an easy-to-manipulate fashion. The kitting representation makes use of an OWL ontology to represent the robots, kit trays, parts, etc., and the actions that are required to construct a kit. The ontology is augmented by a reasoning engine that can construct plans for creating instances of kits and can generate generic commands for robots and sensors to actually construct a set of kits. These researchers have also developed a set of performance metrics for the kitting domain [34]. Metrics include the number of parts correctly put in the kit, the number incorrectly placed, the total distance moved in building the kit, the total number of errors of all kinds, and a score based on the time, distance, correctness of placement, and the number of unnecessary commands carried out.

Morris and Haynes [35, 36] developed a formal language to define how parts are to fit together and to guide assembly by a robot. Their assembly by constraint (ABC) approach defines assemblies based on the concept of degrees of freedom (DOF) of components and the reduction of degrees of freedom as components are assembled. The authors derived a component constraint status, based on the degrees of freedom of one component with respect to its mate (three translational and three rotational). The degree of freedom is Boolean, i.e., it either exists or doesn't. Out of a total of 64 possible combinations, they determined that twenty are unique, three are unrealizable, fifteen are applicable to assembly, and the final two represent totally constrained and unconstrained states. Their intention was to use the constraint-based information to provide sufficient data for the robot to generate its own action plan based on its capabilities and the environment.

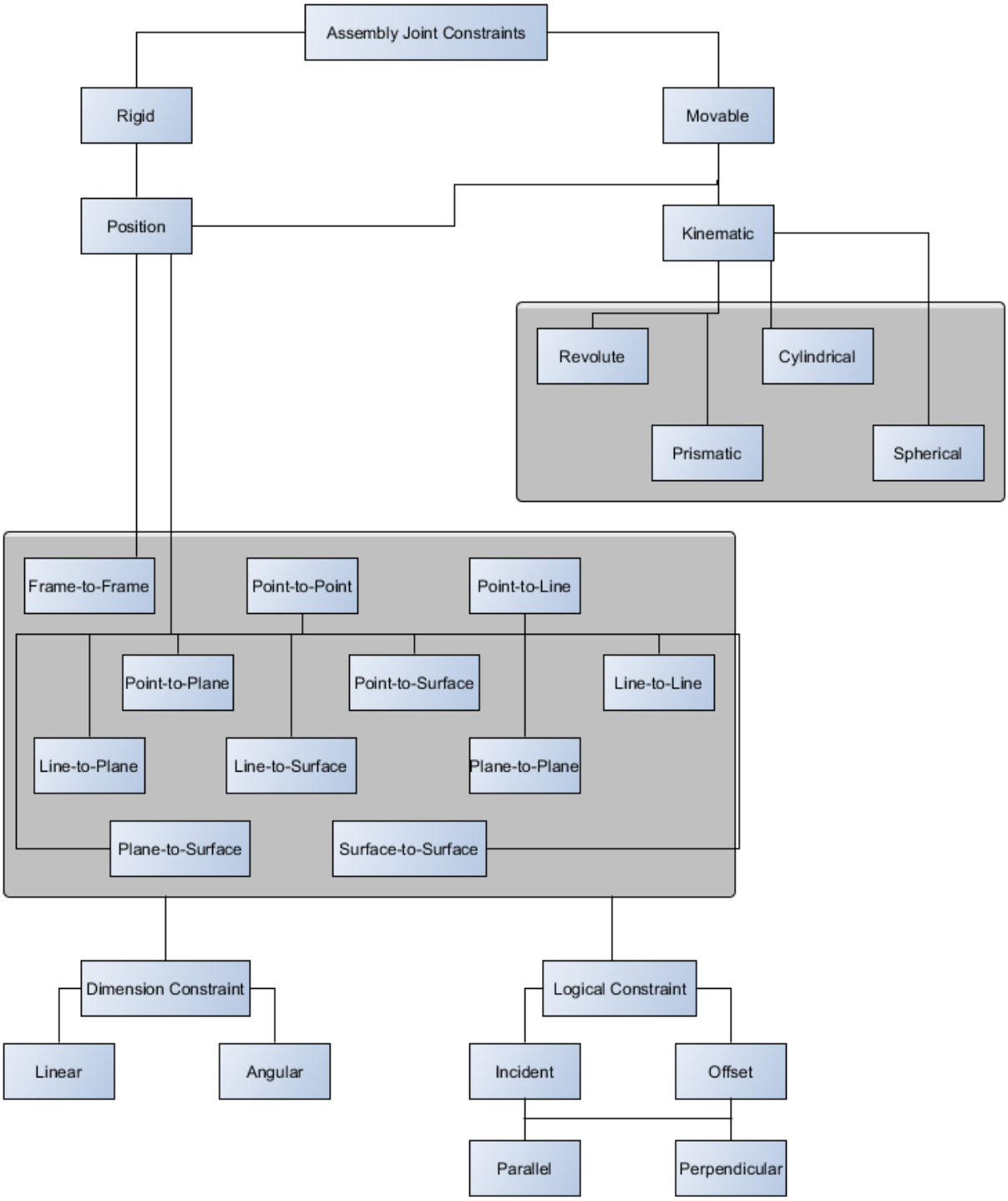

Figure 5. Hierarchical representation of assembly constraints from [32]

Morrow and Khosla [37] extended the work of Morris and Haynes by incorporating specific geometry information when considering robotic manipulation for assembly. Their research

defines MTP (manipulation task primitives), which include a specific geometric interpretation of the relative motion constraints. Rather than list each DOF with a 1 or 0, they combine the translation and rotation DOF for an axis into one symbol which encodes the translation/rotation classification. A resulting 3-tuple represents the DOF of the task frame (permutations of the same three symbols have the same meaning). Morrow and Khosla also developed a set of sensorimotor primitives [37], which are an encapsulation of sensor processing and action (in this case, executing a trajectory) that can form the foundational capabilities for task planning and execution. For example, force primitives are able to utilize the degrees of freedom information to guide their control scheme.

Wu and Kim [38] also propose a variant of constraint-based mating classification. They include geometry considerations (e.g., inserting a round peg has different symmetries than inserting a prismatic peg) for their Structured Assembly Coding System (SACS). This coding system provides an additional geometric constraint definition with the intent of supporting synthesis of compliant control strategies for robot fine assembly motions.

Related to constraint-based representations is the adoption of screw theory for assembly mating. The motions that a rigid body can undergo, or the forces and moments that are exerted upon it can be represented by a screw. Twists represent motion through angular velocities and translational velocities, while wrenches capture the forces and moments that a joint can resist. Konkar [39] developed a representation of assembly mating features based on screw theory and used this approach to determine the relative degrees of freedom between parts in an assembly.

4. Relevant Prior Work in Performance Metrics

Performance metrics for robots have been developed both formally through standards and informally by researchers who needed to evaluate their systems. The formal performance standards for industrial robots (ISO 9283 [40], ANSI/RIA R15.05 [41-43]) are old and cover only limited aspects of performance, mainly relating to point-to-point repeatability. In the area of response robots, however, there has been a concerted effort to develop performance standards [44]. Some of these may be of value for manufacturing applications as well.

Ceballos, et al. [45] define a number of metrics for evaluating navigation algorithms for mobile robots. They include security metrics that measure how well the vehicle keeps a safe distance from obstacles, dimension metrics that measure the length of the chosen path, and smoothness metrics that take into account changes in direction due to re-planning as well as the desire to reduce energy use. The metrics are demonstrated in simulation by showing the results of two control algorithms driving a robot through a maze.

Frommberger, et al. [46] define performance measures for mobile surveillance robots operating in a warehouse. They describe two types of metrics, relating to optimizing the logistics aspects of the tasks and to the efficiency of the surveillance robots. Logistics efficiency is a function of delivery

times, storage efficiency, stock turnover, etc. Robot efficiency depends on mapping and localization, collaboration between robots to cover the area, etc. Since both problems have multiple criteria, the optimal solution is dependent on the needs of the operator of the warehouse. The authors suggest using histories of the observation of the goods flowing through the warehouse to provide a good estimate of how well the task is executed.

Marvel and Falco [47] define a set of force control metrics and a set of force-based assembly metrics. Force control metrics include settle stability, which measures how long the system takes to stop moving when it impacts a surface, obstruction stability, which measures the system's performance when trying to push past an obstacle in its path, control switch stability, which is a measure of how smoothly the system switches from one type of force control to another, surface cohesion, which evaluates how well the system is able to maintain a constant force as it moves across a surface, and incurred force limitations, which measures how well the system deals with extraneous forces while not exceeding its force limits. The force-based assembly metrics are specifically oriented towards measuring how well the system completes a range of assembly tasks. They include the time to complete the assembly, the success rate when attempting a set of assemblies, and the average amount of force needed to complete the assemblies. A set of components, all of which fit into a base, was proposed for computing the metrics in a standard way. The base includes force measurement sensors to provide an independent ground truth measurement.

Singer and Akin [48] provide a survey and categorization of performance metrics for human-robot teams. The metrics are first presented in isolation and then a number of suggestions are made as to how to combine metrics to provide an overall evaluation of a given human-robot task. Individual metrics are divided into task-specific metrics, offline versus real-time metrics, metrics for the level of autonomy of the robots, situation awareness metrics, and communication metrics. Combining the metrics may be done by a simple or weighted summation or by a more structured approach such as using state transition networks. The authors caution that there is a lot of subjectivity in selecting the metrics to use and the way to combine them and this can greatly affect the resulting measures. Errors in measurement can also have significant influence on the final results.

A paper by Burke, et al. [49] describes a method of evaluating the performance of human-robot teams in a search and rescue environment. It involves an intensive analysis of videotaped records of operator-robot teams engaged in typical activities. The analysis addresses a range of issues including communications, developing situation awareness, and the ways in which the humans interact with the robot. No performance measures extracted from this data are described.

Steinfeld, et al. [50] define a set of factors that need to be considered when evaluating a human-robot task and suggest metrics for each factor. The factors include navigation, perception, managing the task and human-robot interaction, manipulation, and social factors (how well the human and robot interact). Other factors that influence the results include communications, robot response times, and human task load. The individual metrics for the different areas also have to be

integrated into a task-level measure. This integration includes some quantitative measures, but also requires subjective evaluation.

For human-robot interaction applications, Fässberg, et al. [51] stress the need to address both the medium of presentation used for communicating information, such as paper or a computer monitor, and the content of the information (i.e., what is selected to be conveyed and when). Each should be optimized for the task and level of experience of the human participants.

Patel and Sobh [52], present a literature review of performance measures for manipulators. They discuss the definition, classification, scope, and limitations of many performance measures for manipulation and include an extensive bibliography. Manufacturing manipulation and grasping dexterity measures based on an industry-focused workshop are detailed in [53, 54].

Shi and Menassa [8] discuss requirements for flexible robotic assembly for a wheel and tire load moving assembly line. They specify three key performance evaluation areas:

- Manipulation skills: peg-in-a-hole assembly in 3D space; Contour match assembly in 3D space; Surface match assembly in 3D space.
- Perception Skills
- Robustness

Design for Assembly and other approaches related to assigning relative costs to assembly operations, which were summarized above in Section 2.2 can also be considered proto-performance metrics for robotic assembly.

5. Performance Metrics Based on a Taxonomy of Assembly Tasks

The previous work indicates that there is a need for a taxonomy that includes the actions and capabilities required for robotic assembly, and that other areas must also be included. These include communications, coordination, time and space considerations, and human factors for applications where humans and robots collaborate. This is a large domain to cover and emphasizes the need, also evident from previous work, for both individual performance metrics for each component of the taxonomy and measures that give an overall estimate of how well the entire system will function.

From the above summary, it can be seen that there are a lot of similarities between the various taxonomies that have been defined for robotic assembly. In this paper we adapt one of the more recent taxonomies, that defined by Huckaby and Christensen [22], with additions from several of the other taxonomies defined in Section 3.1. Because it was designed for manufacturing assembly operations, the Huckaby and Christensen taxonomy is well suited for organizing the performance metrics for subcomponents of assembly. For each element of the taxonomy, we select specific metrics and we also adopt metrics for larger components of the taxonomy (sub-trees) and for the

application as a whole. We assume that each individual assembly action is binary: i.e., it entails joining two distinct parts or subassemblies.

These proposed performance metrics are augmented by consideration of the specific types of assembly join operations, constraints, and associated difficulty scales discussed in the previous sections. These additional relevant conditions and constraints should be taken into account during the design of methods (including test artifacts and procedures) of capturing performance with respect to the metrics. Initially, a static environment is assumed; the metrics and test methods can be extended to cover motion and tracking of an assembly line. Other considerations, such as ease of programming the robot to perform a new assembly, or general human-robot interaction concerns are not covered in this analysis.

5.1. Actions

Figure 6 shows the set of assembly actions defined in the taxonomy. They include actions that accomplish sensing, motion, positioning, component modification, and coordination. Metrics and performance measures can be defined for each of the actions, although some of the actions, such as detect, fasten, and coordinate, are the roots of subtrees whose nodes also need performance measures.

5.1.1. Detect

The Detect subtree (Figure 7) deals with identifying and locating objects that will be manipulated during the assembly. Typically, image-based sensing systems locate features in the world that may be objects, parts of objects, or fixed elements such as fiducial markings. This sensing is also used for visual servoing (guiding the robot to its destination using vision) and to check that each step in the assembly process is carried out correctly. The metrics in Steinfeld, et al. [50] that are relevant to assembly are of value for visual detection. They include measures for passive perception and for active perception. Passive perception interprets the sensor data without active control of the sensors. It is the most common form of perception for manufacturing. Placing a camera on the robot or on a pan-tilt head enables active pointing and searching for objects, which is called active perception.

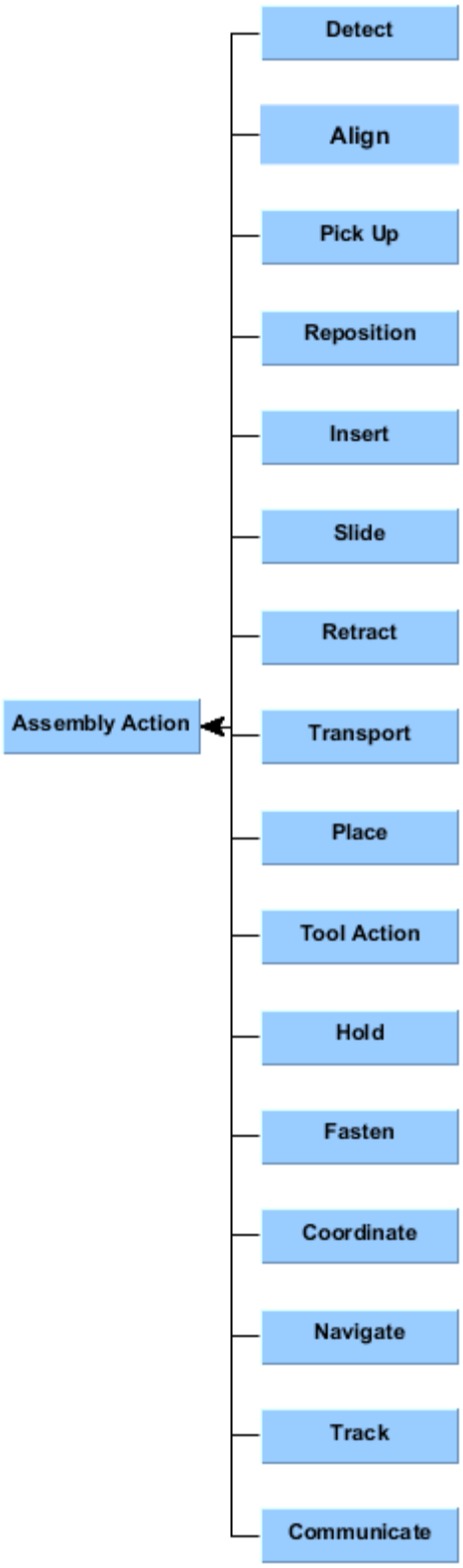

Figure 6: The actions defined in the expanded Huckaby-Christensen taxonomy

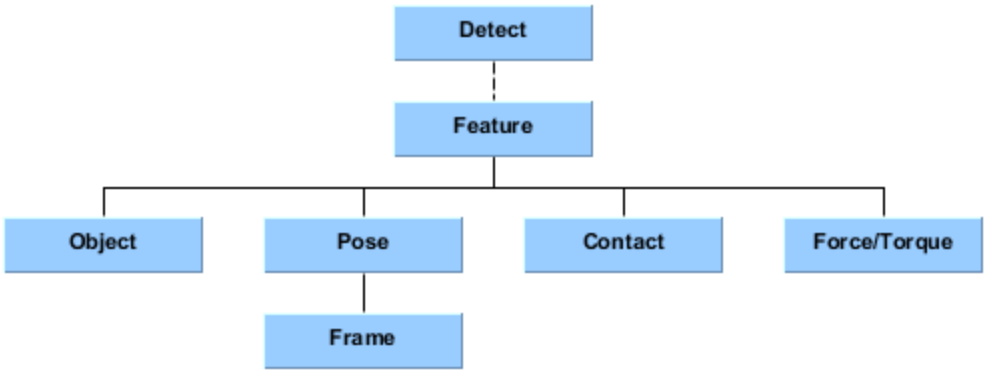

Figure 7. The detection subtree from Huckaby and Christensen

Metrics for passive perception include detection measures such as the number of objects correctly detected and the poses of the objects detected as compared with their true poses. In manufacturing, sensing is often used for in-process inspection, so detection of incorrect objects or defective parts should also be included in the measures. Other measures include how well the system can deal with clutter and occluded parts. For systems that expect parts to be in known locations, the amount of variation that the system can handle can also be used as a performance measure.

For active perception, the goal is usually to find an object that is not in the field of view of the sensors or to guide the robot to complete some action, such as mating two parts or inserting a peg in a hole. Performance metrics include the amount of search needed to find a part and how accurately the position of the part is determined (especially when the sensor is moving). The update rate of the sensors may also be of interest in these cases.

While vision-based sensing covers the object and pose aspects of the Detect taxonomy, force-torque sensing is used for the contact and force/torque aspects. Sensing of force and torque for insertion, part manipulation, or surface following tends to be more *ad hoc*, with specialized algorithms being implemented for different actions. This makes it harder to evaluate how well force-torque mediated actions are accomplished. The metrics in Marvel and Falco, [47], are useful for these actions. The more general metrics in Patel and Sobh [52] are also relevant. Metrics for force-based assembly include the time taken, the success rate, and the maximum and average force used as compared with the desired force. For operations like applying adhesives, spray painting, or welding, surface-following metrics are important. These include the average distance from the surface, the average force applied to the surface, and how well the required area was covered.

5.1.2. Align

This is a compound action, comprised of detect and transport. The Align action requires the robot to detect the object that is the focus of the command and to (visually) servo to the required pose relative to the object. Metrics for this action include how well the robot reaches the required pose (measured in comparison to ground truth) and how long it takes to settle at the required pose.

Parameters that would influence test method design include what object(s) are to be aligned, how many degrees of freedom are involved (i.e., what are the constraints in the goal configuration), and what the required alignment tolerances are. Another aspect to consider is how accessible the align target is, both for the robot's sensors and for its manipulator.

5.1.3. Pick up

Similar to Align, Pick up is also a compound action and requires detecting the object to be acquired, servoing to it, using the same metrics for success, as well as grasping it. The Pickup action requires the ability to detect that the object is in the gripper, perhaps the capability to measure the grip force and the pose of the object in the gripper, and to determine if the object is held firmly or may slip. Settling time may also be an issue if the object must be stationary before being moved.

Variables that influence the test method design are the properties of the object to be picked up (including geometric features, size, material properties, etc.) as summarized in Table 1.

5.1.4. Reposition Object (in Gripper)

There are instances where, after being picked up, an object must be reoriented in preparation for the next task while in the gripper or hand. This requires in-hand manipulation by the robot or the use of external jigs or fixtures to assist in the re-positioning. Metrics include the successful completion rate, time to complete, and whether or not additional hardware is required.

5.1.5. Insert

Assuming the object to be inserted is in the gripper and is aligned with the hole, the Insert action is typically primarily force-mediated. Metrics include the minimum clearance ratio that can be achieved, time to achieve the insertion (which may require a search), the maximum and average forces applied, and the success rate. Variables include the types of geometries involved for both the target part (e.g., hole, slot, with and without chamfer) and the inserted part (cylinder, prismatic part, …), insertion direction, and accessibility of the insertion location, as summarized in Table 1.

5.1.6. Slide

Slide actions also typically make use of force sensing to maintain contact and to ensure that not too much force is applied to the surface. Metrics include the accuracy of the path compared to the commanded path, the average and maximum forces applied, the accuracy of start and stop positions, and the percentage of time that the surfaces are in contact during the Slide motion. Variables include the types of geometries and properties of both objects, as well as the number of constraints on the sliding motion (single or multiple surfaces).

5.1.7. Retract

Retract can be considered the complement of Place. Metrics for Retract include the accuracy of the path taken and of the stopping point, and the time required for execution. Factors influencing test method design include the tolerance associated with the path, how obstructed the retract path

may be, whether intermediate points must be specified explicitly, or the robotic system can autonomously calculate a collision-free path.

5.1.8. Transport

Transport may include actions carried out by a robot within its work volume or by a vehicle moving between workstations. Metrics include the accuracy or repeatability with which the robot moves to the start position, the maximum deviation from the planned path between the start and end positions, and the accuracy or repeatability with which it reaches the goal position. Transport actions can also include obstacle recognition and avoidance, for which metrics include the accuracy of determining the position and size of the obstacle, the success rate of planning paths around it, and the success rate of executing the planned path without hitting the obstacle or other objects. The time taken for the Transport action can also be used as a metric.

Transport in this taxonomy implies larger distances, and hence corresponds to the Gross Motion described in Section 2.1.

5.1.9. Place

The Place action has similar metrics to Transport, but may include tighter constraints on the final pose. Place corresponds to the Fine Motion discussed in Section 2.1. This may require visual or force servoing as in Insert. Metrics would therefore include maximum allowable forces, as well as positioning and velocity tolerances. Factors to be considered include the direction of approach, accessibility of the goal location, and the number and type of geometric constraints to be achieved (e.g., placing a planar surface onto a planar surface versus having to ensure multiple surfaces are in contact). The Place action is one that does not involve insertion of one part into another.

5.1.10. Tool Action

A tool in this context could either be an end effector that is attached to the robot or a separate tool, such as a screwdriver, that is picked up by the robot's gripper or hand. The ability to use a tool requires that a tool frame be defined and that the robot be able to pick up and activate the tool. Picking up the tool may require Align and Pickup actions, which have their own metrics. Using the tool requires ensuring the offset and pose of the tool are maintained correctly and the action is carried out correctly.

5.1.11. Hold

Hold requires the robot to maintain a specific pose for a specified amount of time. Metrics include the accuracy with which the pose is acquired, the maximum and average deviations for the specified pose, and the length of time the pose is held relative to the specified time.

5.1.12. Fasten

Fasten is a class of actions including those shown in Figure 8. In addition to the ones in the Huckaby taxonomy, snap or press fitting and nut-and-bolt fastening are key methods that must be considered. Other fastening actions might be defined for special cases, such as applying adhesive tape, wrapping, stitching, etc. Each may have its own metrics, but in general they will be similar

to those for previously-described situations such as Sliding, Tool Actions, etc. Fastening is central to assembly and has its own taxonomies (e.g., Ziout and Azab [55]).

Figure 8. The Fasten subtree from Huckaby and Christensen

5.1.13. Coordinate

The Coordinate action refers to a class of constraints, shown in Figure 9. Metrics clearly are time-related: Did one action start before another if that was the intent? Did the actions occur simultaneously?

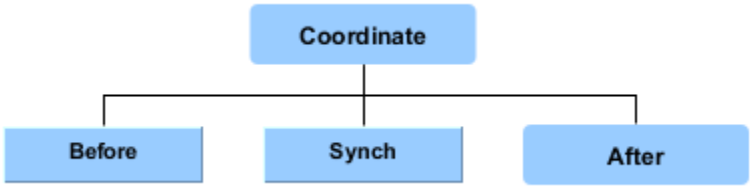

Figure 9. The Coordinate subtree from Huckaby and Christensen

All the actions are subject to constraints, some of which are shown in Figure 10. The constraints usually set the parameters for the metrics. For example, an action may have to take place within a given duration. In this case, the metric should use this duration in determining how well the action was completed. Similarly, the motions, poses, speeds, etc., provide the bounds within which successful actions can be completed and their performance measured.

1.1.1. Navigation

Navigation is a significant task within a factory that uses mobile robots and corresponds to the Gross motion discussed above in Section 2.1. Metrics include percentage of navigation tasks successfully completed, amount of deviation from the planned route (e.g., distance traveled compared to length of planned path), percentage of obstacles successfully avoided, and time to complete the action.

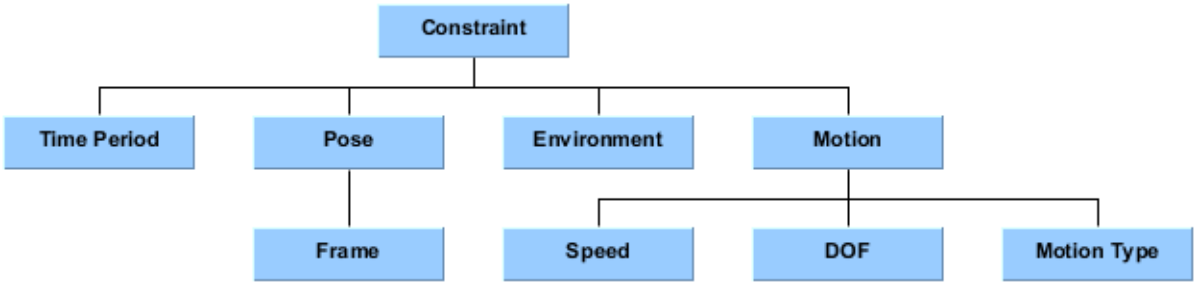

Figure 10. The Constraint subtree from Huckaby and Christensen

1.1.2. Track

Related to coordination and navigation is the tracking task. This is relevant when the assembly operations are occurring on a moving line. The robot has to detect, align, pick up, insert, fasten, and perform other tasks while the line is moving. The metrics for this are speed and accuracy.

1.1.3. Communications

There are two kinds of Communications metrics for robotic assembly. One has to do with the medium used, such as wireless Ethernet, while the other has to do with what is actually communicated. For the medium, standard metrics include bandwidth, latency, and jitter (variance in latency). The data sent over the medium can vary widely depending on whether the application is run using a centralized or distributed architecture. Measurements for what is sent over the medium include numbers of messages of different types per task, total number of bytes per task, average and maximum data rate during the task, and perhaps saturation measures such as delays due to unavailability of the network.

1.1.4. Performance Metrics Summary for Key Actions

Table 2 through Table 11 summarize the parameters and metrics for a subset of the actions discussed above. The actions selected for detailing are those that are likelier to be implemented in the near-term, and for which there exists a set of performance expectations. For example, the "Collaboration" action requires further research and definition of what the metrics and parameters should be. Some of these are discussed in the next section. As can be seen in the tables below, there are recurring requirements, such as the need to evaluate the pose accuracy for a part. For this, an existing standard from ASTM may be used [56]. Time to accomplish a task is also a universal metric. For any performance metric, measurements must be repeated a sufficient number of times to achieve statistical significance.

Table 2: Detect action

Action	Property/Parameters	Metric/Test Method
Detect	Object or Feature (e.g., hole)	- Object ID is Correct - Context: occluded (Y/N); in bin (mixed parts (Y/N) - Distance at which initial detection occurs - Minimum object or feature size detected
	Object/Feature Pose	- Pose accuracy (ASTM E2919-14)
	Contact	- Touch sensitivity - Stability
	Force-Torque	- Settle stability
	Cycle Time	- Time to detection

Table 3: Align action

Action	Property/Parameters	Metric/Test Method
Align	Object; Target Feature to Which to Align	- Object ID is correct - Target feature ID is correct
	Target Object Pose (see Fig. 3 for list of potential constraints), Degrees of Freedom (e.g. Radially symmetric or not); Tolerance	- Accuracy of object pose with respect to target
	Accessibility of Target Pose (constrained/unconstrained)	- Ability to achieve; - Number of degrees of freedom constrained; - % of access volume constrained.
	Cycle Time	- Time to Align

Table 4: Pick Up action

Action	Property/Parameters	Metric/Test Method
Pick Up	Object	- Object ID is correct
	Object Properties (friction, rigid/non-rigid, geometric features of relevance)	- Ability to acquire object - Maximum weight - Minimum and maximum dimensions

		- Minimum friction - Grasp stability
	Constraints (e.g., required grasp points)	- Correct grasp points
	Settling Time	- Settle stability
	Cycle Time	- Time to acquire object

Table 5: Transport action

Action	Property/Parameters	Metric/Test Method
Transport	Object	- Object ID is correct
	Start and End Poses	- Pose accuracy (ASTM E2919-14) - Robot accuracy (ISO 9283) - Robot repeatability (ISO 9283)
	Contact	- Touch sensitivity - Stability
	Force-Torque	- Settle Stability
	Object Properties (friction, rigid/non-rigid, geometric features of relevance)	- Ability to hold object in transit - Maximum weight - Minimum/Maximum dimensions - Minimum Friction - Grasp stability
	Path Properties (velocity, distance, occluded, constrained)	- Max velocity, max acceleration - Obstacle detection and avoidance; Accuracy of: o Obstacle classification or identification o Obstacle pose o Path replanning to avoid obstacle - Do intermediate points have to be specified? - Maximum deviation from programmed path - Success rate for achieving path
	Cycle Time	- Time to transport

Table 6: Reposition action

Action	Property/Parameters	Metric/Test Method
Reposition	Object	- Object ID is correct
	In-hand Target Pose	- Object pose accuracy
	Object Properties (friction, rigid/non-rigid, geometric features of relevance)	- Ability to reorient Object - Maximum weight - Minimum and maximum dimensions - Minimum friction - Grasp stability - Additional hardware required (intermediate fixture for re-orienting)
	Contact	- Touch sensitivity - stability
	Force-Torque	- Settle stability
	Cycle Time	- Time to reposition object

Table 7: Place/Retract action

Action	Property/Parameters	Metric/Test Method
Place and Retract	Object Target Feature (Plane)	- Object ID is correct - Target feature ID is correct
	Target Pose; Degrees of Freedom	- Object placement pose accuracy (ASTM E2919-14)
	Contact detection; Max allowable forces	- Touch sensitivity - Stability
	Force-Torque	- Maximum force exerted during placement - Settle stability
	Placement Path Parameters (distance, velocity)	- Maximum velocity, maximum acceleration - Obstacle detection and avoidance - Obstacle classification or identification - Obstacle pose - Path replanning to avoid obstacle - Max deviation for programmed path - Success rate for achieving path

	Direction of approach	- Difficulty of approach direction (typically, top-down is easiest; bottom-up is generally hardest)
	Cycle Time	- Time to complete placement

Table 8: Slide action

Action	Property/Parameters	Metric/Test Method
Slide	Object to Slide Surface to Slide Upon	- Object ID is correct - Surface ID is correct
	Start and End Poses	- Pose accuracy (ASTM E2919-14)
	Slide Trajectory; Degrees of Freedom (see Fig. 3 for list of potential constraints, e.g., anywhere along plane or constrained to follow a line on plane)	- Accuracy of path
	Contact	- Touch Sensitivity - Stability - Percentage of Time that Surfaces are in Contact
	Forces	- Minimum and Maximum Forces
	Cycle Time	- Time to Complete Sliding Action

Table 9: Insert action

Action	Property/Parameters	Metric/Test Method
Insert	Object To Insert Destination Feature in which to insert	- Object ID is correct - Feature ID is correct
	Object and Destination Receptacle Properties (friction, rigid/non-rigid, geometric features of relevance such as chamfers)	- Grasp stability - Minimum clearance ratio
	Contact	- Touch Sensitivity - Stability
	Insertion Forces	- Minimum force

		- Maximum force
	Direction of approach	- Difficulty of approach direction (typically, top-down is easiest; bottom-up is generally hardest)
	Cycle Time	- Time to complete Insertion

Table 10: Hold action

Action	Property/Parameters	Metric/Test Method
Hold	Object	- Object ID is correct
	Object Properties	- Maximum weight
	Hold Pose	- Pose accuracy (ASTM E2919-14)
	Hold Time	- Length of time pose is held versus required time
	Contact (if required)	- Touch sensitivity - Stability
	Force-Torque	- Settle stability

Table 11: Tool action

Action	Property/Parameters	Metric/Test Method
Tool Action (may require Align, Pickup, Retract, or other Actions); Includes Drilling	Tool Type	- Tool identification is correct
	Position	- Position accuracy (ASTM E2919-14)
	Tool Command (Attach, Detach, Start Tool, Stop Tool, etc.	- Command completed correctly
	Contact	- Touch sensitivity - Stability
	Force-Torque	- Settle stability
	Cycle Time	- Time to complete individual tool action(s) - Total time to complete series of tool actions(s), e.g., including move to tool

		changer, attach, detach, and perform operations with tool

1.1.5. Global Task Metrics

Global task-level metrics include time to completion, percentage of assemblies correctly completed, percentage of errors corrected, percentage of bad parts correctly found and rejected, and amount of slack time (for example, in switching from one task to another or waiting for parts to arrive). Some measure the Process Capability Index. Process Capability indices relate the process mean and standard deviations to the Upper and Lower tolerance limits defined for an assembly. Typically, a given process is compared to $\pm 3\sigma$ of a normal distribution.

For tasks in which humans and robots collaborate, another set of global metrics can be defined. They include measures of the task load on the human and robot, how well the human and robot share the tasks (e.g., how long each spends waiting for the other as a percentage of total task time or total number of task elements), amount of explicit communication required to accomplish the task, and how well the task allocation takes advantage of the capabilities of each of the participants.

Kitting is a specialized subset of assembly, wherein parts are delivered to the assembly station in kits that contain the exact parts necessary for the completion of one assembly object. The kit itself must be assembled, although the constraints on the insertion of the parts into the tray are typically much looser than for the assembly join operations themselves. Nevertheless, it has been identified as a fruitful area to begin robotic integration into assembly operations. Balakirsky et al. have defined metrics for kit building [57]. The static kitting metrics range from tallies, such as number of commands executed and the execution time, total distance moved, and total objects moved, to errors of various sorts, such as asking the robot to move a part that is too heavy or outside its work volume. The authors introduce additional execution metrics, such as manipulation robustness, transporting ability (how well a part is moved by the robot), contact errors (number of collisions between the robot and objects), failures during planning, and during execution, including how well the system is able to recover from failures during execution. This latter set, along with some of the static kitting metrics, can be adopted for assembly operations beyond kitting.

Robots must function in the presence of uncertainty, so robustness is a key global metric. An assembly operation is called robust if its performance or execution is substantially insensitive to variations that might occur, for example, in the sizes, shapes, and locations of parts, external loads, or operating conditions, so long as the variations are within the specified tolerances (adapted from Whitney). The robustness of each individual assembly operation discussed above can be evaluated by configuring the test method to present parts with size variations at the extrema of the tolerance zones and locations, and by stressing other environmental factors, such as the lighting conditions if visual sensing is being used, or the speed of the assembly. Donald [58] examines error detection and recovery abilities in the presence of uncertainty from sensing errors, control errors, and errors in the geometric models of the robot and the environment. The robustness of the

robotic system to the accumulation of variations in the individual assembly operations can therefore also be assessed.

2. Representing Robot Capabilities

Once the performance of a robot executing an assembly task is determined, the performance needs to be represented in such a way that it could be of use to the end user. In the simple case, a spreadsheet or database could be put together to allow the end user to look up whether a robot can or cannot perform a given task. However, these robot capabilities could also play a role in planning the assembly of a product when multiple tasks are needed to perform the assembly operation. There may also be multiple robots that could perform a given task, and it is important to know which robots are available and have the capability to perform the needed operations. If this information were encoded in a computer-interpretable knowledge representation that could be fed into a planning system, the planning process could be partially, if not fully, automated.

In this context, we refer to a robot capability as the ability of a robot to perform a specific action on a specific object or set of objects. Therefore, the robot capability is specific to the individual object(s) and the individual action. This is different than the type of information that one would find on a robot specification sheet, which could include general characteristics such as how much a robot can lift or the reach of a robot. While such information is important and could be included in a robot capability model, it is not the focus of this paper.

One of the challenges of defining capabilities in this way is that there are an almost infinite number of combinations among robots, actions, and objects, and it would be impossible to test all of these combinations. The hope and expectation is that the test methods that are being developed will provide a strong representative set that could be used to extrapolate to other similar manufacturing assembly situations. The design of these test methods is a science in itself to ensure that they are both representative of the manufacturing assembly domain and allow the performance of the robot to be predicted in similar situations.

To better understand the use of the robot capability model, one can imagine the following scenario. A company has a new product that they want to assemble. They have a large number of robots available to them on the shop floor, and need to determine how to assign robots that have the required capabilities to individual tasks in the assembly process. Each assembly process is comprised of a large number of tasks, so they need to optimize the sequence of operations and associated resources (the process plan) to accomplish the assembly. To do this, we will assume that they have a software tool that imports the following information:

- Robots that are available
- The capabilities of those robots
- The desired end state (i.e., the final assembly)
- The objects that compose the final assembly and their pertinent characteristics

The output of this tool would be a process plan that assigns robots to the tasks needed to accomplish the assembly.

In this rest of this section, we will explore the information requirements necessary to represent robot capabilities in a computer-interpretable format.

2.1. Robot Capability Model Information Requirements

As described above, we define a robot capability by the intersection of a robot, an action, and objects in the environment. Therefore, we need a way of representing each of these items, including a way that we can put them together into an instance of a robot capability.

It is envisioned that both the action and the robot will need to be represented at various levels of abstraction. In the case of a robot, one may want to associate capabilities with the robot as a whole, the arm of the robot, or maybe only the gripper of the robot. As such, we will need a way to represent the composition of the robot and have the ability to represent the capabilities at each level.

Similarly, actions will need to be modeled at various levels of abstraction, as described in the taxonomy. For example, one may want to associate a robot capability with an overall assembly action, or perhaps to only one step in an assembly action (e.g., attach Part A to Part B), or even to a sub-step in the "attach" step (e.g., pick up Part A).

In general, we envision a construct such as the following to bring the various pieces together to model the robot capabilities:

- Robot Capability:
  - Robot Composition: pointer to a robot concept at an appropriate level of abstraction
  - Activity: pointer to an action concept at an appropriate level of abstraction
  - Object(s): pointer to one or more objects (and their associated characteristics) that the action affects
  - Capability: a representation of how well the robot can perform the action on the object. This could be a Boolean (yes/no), a probability, or some other form of representation.

The next sections provide a literature review of the top three bullets above and describe in more detail the information requirements for each.

2.2. Representing Robot Composition

An important role of representations of robot capabilities and performance is their support for composing a complete robotic system from individual components. This *composability*, or ease with which a system can be put together with components to meet functional requirements, requires that each component provide documented interfaces that enable them to be identified, configured, and used without resorting to trial-and-error procedures. If standards for these

interfaces are widely implemented, those components display a high degree of *interoperability*, making it much easier to compose a system from them.

Understanding the behavior of a system of components, given an understanding of their behavior and the rules by which they are put together, is *compositionality*. Qualitative system behavior can be determined from the descriptions of nominal component behavior, but quantitative prediction of system behavior of actual components requires that measures of their performance be included. Performance measures could include the results of tests, calibrations, or statistical distributions of observations over time. Analytical prediction of behavior is difficult, and simulation is often used as a method for understanding the functional behavior and performance of a system under relevant scenarios. Developing a convincing simulation framework is also a difficult task, but recent advances in computing have made it possible to simulate systems at even the physics level in close to real time [59].

2.2.1. Related Efforts

Many systems and standards for robot component interfaces have been developed. The Robot Independent Programming Environment (RIPE) and Language (RIPL) [60] formed an object-oriented distributed robot control system developed in the early 1990s that employed programming concepts such as inheritance and polymorphism to achieve software application code portability and reuse. Researchers from Sandia National Laboratories implemented RIPE/RIPL in a series of applications focused on robotic nuclear waste handling and remediation. Although not currently supported, RIPE and RIPL influenced later robot software efforts with its distributed heterogeneous architecture and application of modern object-oriented design principles.

In the late 1990s, the U. S. Department of Defense sponsored a joint program that brought together its major customers of autonomous mobility systems to define an architecture for interoperability among platform components and software applications. This effort was originally known as the Joint Architecture for Unmanned Ground Systems (JAUGS), later shortened to JAUS to reflect its expansion beyond purely ground-based systems [61]. Although focused on mobile robots, parts of the JAUS specification applied to manipulators used for applications such as explosive device neutralization. JAUS has been standardized through the Society of Automotive Engineers (SAE) in a series of numbered standards from the AS4 Unmanned Systems Technical Committee. An open source implementation, OpenJAUS, is available [62].

In 2000, robotics researchers at the University of Southern California developed an open-source robotics framework, Player, that implemented a client-server software framework that allows distributed control of robots in a networked environment [63, 64]. Player is a set of software application programming interfaces (APIs) with bindings to many popular languages such as C, C++, Java, and Python, and a collection of implementations for robots, sensors, and other devices that support plug-and-play connectivity and portable programming. The Player project also includes simulators for a 2-D world, Stage, and a 3-D version, Gazebo.

Also in early 2000, the European Union sponsored a project proposed by the European Robotics Network EURON to develop an open robot control software framework, called OROCOS [65, 66]. Championed by Herman Bruyninckx of the Katholik University Belgium, the project has grown to include the original core robot control functionality, as well as toolkits for kinematics and dynamics analysis, Bayesian filtering, and general real-time control.

In 2007, the technology incubator Willow Garage sponsored a project to build well-tested implementations of robot control software written at nearby Stanford University. The project, known as the Robot Operating System (ROS) [67, 68], follows a distributed open source model with core facilities for defining messages and services and establishing distributed networked communication among components of a complete robotic system. Contributors are encouraged to provide packages back to the ROS community, ranging from device drivers for sensors to complete applications.

Around the same time, in 2008, the Association for Manufacturing Technology sponsored a project at the University of California Berkeley to develop a standard interface to manufacturing equipment. This project, MTConnect, was built upon an Internet web framework using the hypertext transfer protocol (HTTP) and the extensible markup language (XML) to exchange machine-readable information about the properties and operating condition of manufacturing equipment [69]. Similar to a related and popular standard for industrial automation connectivity, Open Platform Connectivity (OPC), MTConnect took the additional step of standardizing the so-called ‘tags” or names for specific information items. This level of standardization permitted portable client applications to connect to any MTConnect server (called agents) independent of vendor. Typical client applications include mobile human-machine interfaces for observing machine tool operating status, or historical data loggers to identify trends [70].

The Institute for Electronics and Electrical Engineers (IEEE) published a standard, IEEE 1872-2015, Ontologies for Robotics and Automation, that provides a core ontology for robotics and automation (CORA) that specifies general concepts, relations and axioms, together with other ontologies that support the CORA information model [71].

2.2.2. Comparison of Robot Composition Methods

These specifications and standards are all logically divided into information about robot system components (the “what”), and the format of information exchanged between the components (the “how”). Typically, information about the components is needed when they are procured and configured into the larger system, while the information exchange mechanisms are used during operation. For example, information about components may be loaded into a simulation system that does not otherwise employ the communication mechanisms used by the actual components during operation. Likewise, information about the component characteristics may never be exchanged during steady-state operation, only messages for control and status that are focused on the activity rather than the parts that make up the system. However, in more sophisticated use cases, all this information may be used continually in order to dynamically schedule tasks, bring

in mobile assets, and optimize production around the strengths and limitations of manufacturing resources, including people.

It is helpful to compare different methods of representing robot composition in order to understand how they can coexist in a common manufacturing environment. Three of the standards described earlier will be compared: ROS, MTConnect, and IEEE 1872.

ROS provides both a model of devices, primarily robots, and a messaging interface to robots, sensors, and other components in a robotic system. The robot model uses the Unified Robot Description Format (URDF) defined in XML Schema [72]. Robots are composed of links and joints, each with properties that support kinematic and dynamic analysis and control, visual representations for simulation and animation, and determining the potential for collisions. Figure 11 shows a representative sample of a link from [72] indicating the inertial properties needed for dynamic analyses, a simple visual representation, and a bounding region for collisions. More complex links can be defined with references to geometry meshes, such as STL (originally stereolithography) files. These more detailed definitions provide improved visualization and collision avoidance, at the expense of computation time. URDF joints are described similarly, with properties such as friction, damping, and joint limits.

```
<link name="my_link">
  <inertial>
    <origin xyz="0 0 0.5" rpy="0 0 0"/>
    <mass value="1"/>
    <inertia ixx="100"  ixy="0"  ixz="0" iyy="100" iyz="0" izz="100" />
  </inertial>

  <visual>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <box size="1 1 1" />
    </geometry>
    <material name="Cyan">
      <color rgba="0 1.0 1.0 1.0"/>
    </material>
  </visual>

  <collision>
    <origin xyz="0 0 0" rpy="0 0 0"/>
    <geometry>
      <cylinder radius="1" length="0.5"/>
    </geometry>
  </collision>
</link>
```

Figure 11: ROS Unified Robot Description Language (URDF) instance for a robot link. [72]

MT Connect provides a schema for a Device Model, shown graphically in Figure 12, that can be queried using standard HTTP web requests using the unified resource locator (URL) of the device and the well-known path "/probe". A reference implementation is publicly available and continually running at agent.mtconnect.org/probe, and will return an XML-formatted description of the actual device. A sample of this output is shown in Figure 13.

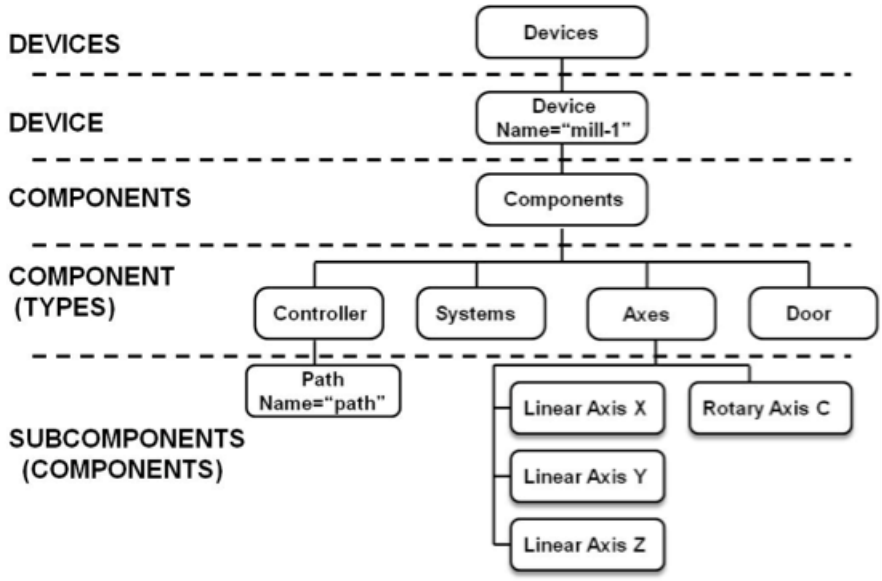

Figure 12: MTConnect Device Model, a graphical depiction showing the hierarchy of component modeling information that is formally expressed in XML Schema. [70]

```
- <Components>
  - <Axes id="ax" name="Axes">
    - <Components>
      - <Rotary id="c1" name="C">
        - <DataItems>
          - <DataItem category="SAMPLE" id="c2" name="Sspeed"
            nativeUnits="REVOLUTION/MINUTE" subType="ACTUAL" type="SPINDLE_SPEED"
            units="REVOLUTION/MINUTE">
              <Source>spindle_speed</Source>
            </DataItem>
          - <DataItem category="SAMPLE" id="c3" name="Sovr" nativeUnits="PERCENT"
            subType="OVERRIDE" type="SPINDLE_SPEED" units="PERCENT">
              <Source>SspeedOvr</Source>
            </DataItem>
```

Figure 13: Sample output of a web query for a running machine, using the `/probe` path. This output shows the spindle speed and speed override capabilities of the machine (a machine tool). [70]

The current operating condition of machines is available via the counterpart path /current. In this case, actual values of current operating conditions are reported, timestamped by the local system.

An advantage of the MTConnect specification is that it defines the required names for each data item. This allows for application and device interoperability, but only around the concepts defined in the specification. Figure 14 shows a list of the terms in the most recent version of the schema. New technologies can be incorporated by extending the schema, with these extensions being shared among the community that is developing the new technology. Once the extensions have been validated, change requests are submitted for the next version of the specification.

AssetCountsType	ActuatorType	ElectricType
AssetCountType	DoorType	DataItemsType
HeaderType	SensorConfigurationType	DataItemType
MTConnectDevicesType	ChannelsType	DataItemConstraintsType
DevicesType	ChannelType	DataItemValueElementType
ComponentType	AxesType	DataItemFilterType
ComponentDescriptionType	AxisType	SourceType
AbstractConfigurationType	LinearType	InterfacesType
ComponentConfigurationType	RotaryType	ReferenceType
CommonComponentType	SpindleType	ReferencesType
ComponentsType	SystemsType	InterfaceType
DeviceType	SystemType	BarFeederType
ControllerType	PneumaticType	MaterialHandlerType
PowerType	HydraulicType	DoorInterfaceType
SensorType	LubricationType	ChuckInterfaceType
PathType	CoolantType	

Figure 14: MTConnect component listing, showing the formal names associated with component definitions. Conforming MTConnect client and server applications share this common understanding of components. A revision process is in place to expand this list as new technologies are supported. [70]

IEEE 1872 provides a high-level model of concepts and relationships in the domain of robotic systems. Higher-level concepts that apply to all domains, such as “quantity,” “proposition,” ”device,” and “group,” are referenced from the Suggested Upper Merged Ontology [73]. Definitions of terms include “robot,” “coordinate system,” “pose,” “robot interface,” and many other terms in the domain of robotics. Relationships between concepts are expressed in the SUMO

knowledge interchange format (KIF). For example, a robot is both a device and an agent, shown graphically in Figure 15.

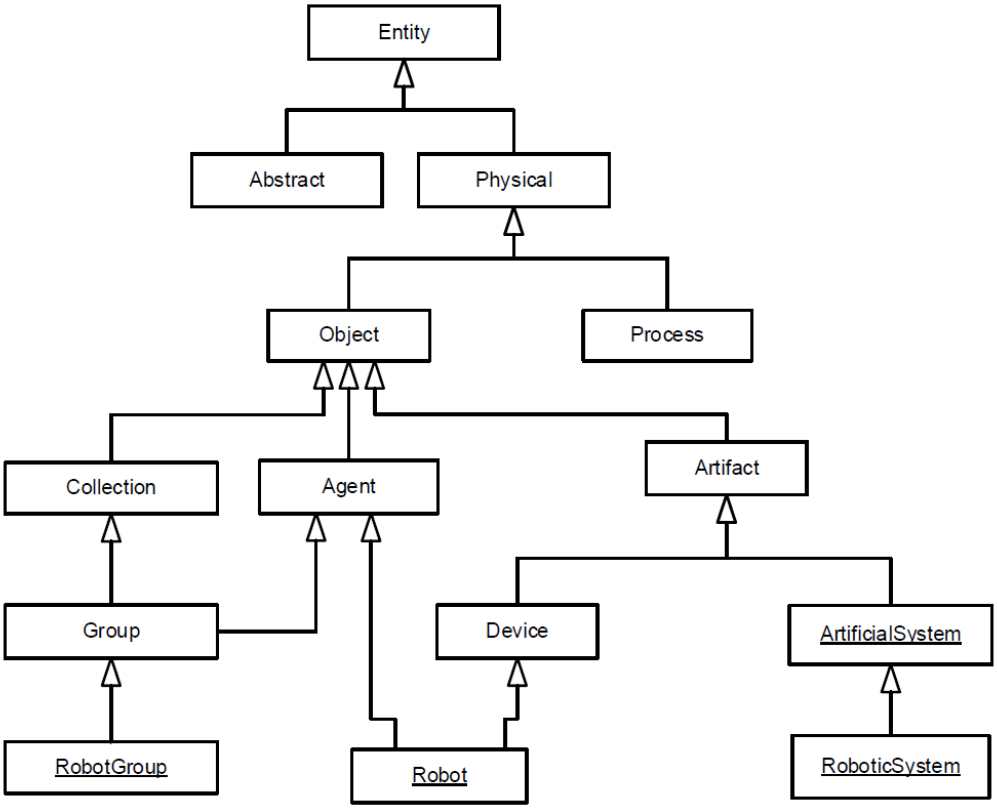

Figure 15: IEEE 1872 ontology, a graphical depiction that shows the relationships between concepts. Here, a robot is both a device and an agent. [71]

2.3. Representing Robot Actions (and associated constraints)

In this section, we will explore the information required to represent robot assembly actions for the purpose of characterizing robot capabilities, as well as provide a brief literature review focusing on how these requirements have been addressed by others. We will focus on three classes of requirements:

- The representation of the action itself with relevant attributes
- The ability to decompose the action into its sub-components to allow one to associate robot capabilities at various levels of abstraction
- The ability to associate constraints with the action

Each of these requirements is described in more detail below along with a review of how they have been addressed in the literature.

2.3.1. Representation of Assembly Actions and Relevant Attributes

The literature review related to the representation of assembly actions and attributes was discussed in Section 3.1 of this paper.

2.3.2. Representation of Action Decomposition

Action decomposition involves breaking down a higher-level action into its constituent sub-actions, which could be further decomposed to multiple levels of detail.

In [29] (referenced earlier), Tan uses a hierarchical task tree to decompose a cable harness assembly. In the tree, the highest level represents the main goal that is meant to be accomplished. This can be broken down into lower level sub-goals that, when combined, achieve the higher level goal. The ordering of these sub-goals is specified using natural language phrases such as "Do 1, then 2, then 3" or "Repeat 1 then 2 for three parts," These sub-goals are further decomposed until they reach primitives that can be directly executed by the system.

In [74], Mosemann and Wahl analyze hyper-arcs of an AND/OR graph that represents an automatically-generated assembly plan. They then use the Unified Modeling Language (UML) to model the robot's tasks and the corresponding skill primitives. The skill primitives are the lowest level commands that are available to the robot to control robot movement (e.g., "MoveTo"), control the gripper (e.g., "OpenGripper"), or task the sensors (e.g., "LocateObject").

In [75], Rosell explores the use of Petri Nets to represent a hierarchy of assembly operations. Specifically, he mentions the use of Hierarchical High Level Petri Nets (HHPN) to allow a system modeler to describe a set of submodels that can all contribute to a larger model by interacting with each other in a well-defined way. This allows a larger assembly plan to be represented as a set of lower-level assembly plans. In this context, a Petri Net place or transition can have a submodel associated with it.

OWL-S [76] (Web Ontology Language – Services) is an ontology of services that allows Semantic Web applications to discover, invoke, compose, and monitor Web resources that offer particular services and have particular properties. While not specifically focused on robots or robotic assembly, it does offer a set of general constructions that could be directly applicable to modeling assembly action decomposition. It has a structure in which high level processes (composite processes) are composed of lower-level processes (simple processes) which are composed of atomic processes. Composite processes correspond to actions that require multi-step actions; simple processes provide an abstraction mechanism to provide multiple views of the same atomic process; and atomic processes correspond to the actions that can be performed in a single interaction. Also of interest in OWL-S is the way that actions can be ordered. A rich set of control constructs is defined, consisting of sequence, split, split-join, any-order, and choice, which allow one to order actions to show how they contribute to the higher level process.

Although the four approaches listed above employ different representation mechanisms (i.e., hierarchical task trees, UML, Petri Nets, and OWL), they all fundamentally contain the same core

set of information. This includes a mechanism to break down a large task into smaller tasks, the concept of the lowest level task being something that can be executed, and a mechanism to order the tasks to address the higher level goal. These concepts will serve as the basis for the development of the future robot capability model.

2.3.3. Representation of Assembly Constraints

The third aspect of robot actions involves representing the constraints that go along with assembly actions. In addition to representing the actions that the robot can perform, we must also represent the bounds and limits in which that action can be performed. As a simple example, a robot may only be able to pick up a certain type of object as a function of how far that object is from the robot arm and the orientation of that object. Constraint-based representation are discussed in Section 3.2 of this paper, and include publications such as Haynes and Morris [35] and Morrow and Khosla [37], which have explored using constraints as a basis for assembly operations.

In addition to the above, Pramanik et al. [77] use a tolerance synthesis scheme to minimize the total cost of manufacturing by considering constraints related to the assembly of a planetary gearbox part as well as constraints related to the functional requirements of the part. Assembly constraints are considered restrictions on the variability of the geometry of the features that would block assembly of the part. The authors construct an assembly graph (connectivity diagram) to determine the assembly constraints. Functional requirement constraints show the parameters that should be optimized to accomplish the function of the part (i.e., converting a set of inputs to a set of outputs). They might include constraints such as reducing friction or maximizing power. The cost functions and assembly and functional requirement constraint equations are generated and are optimized using non-linear minimization tools to find the optimal plan.

2.4. Representing Objects/Environments and Their Characteristics

For the construction of a knowledge representation to be more than an academic exercise, its contents must be constructed so as to serve specific applications. An initial description of an application is needed to develop an initial knowledge representation for the application, but the initial representation must not be regarded as fixed. During the development of an application, it is necessary for two reasons to modify the knowledge repeatedly. First, it commonly occurs that new knowledge requirements are discovered in the course of implementing the application to support the planned functionality. Second, the functionality of the application may change, requiring new knowledge. After an application is mature, changes to the internal structure of the implementation or further changes in functionality (often enabled by new technology) may require further changes in the knowledge representation.

There has been an environment model developed for one sub-domain of assembly [33], so we will focus this section on that sub-domain. For the kitting domain described previously in Section 3.2, the primary application being pursued jointly by NIST and Georgia Tech Research Institute is a fully automated kitting workstation in which a single one-armed robot constructs kits of parts to be used elsewhere in an assembly operation. The knowledge representation for the kitting

workstation is built in XML Schema Definition Language (XSDL) [78, 79] and translated automatically into Web Ontology Language (OWL) [80, 81]. A MySQL database [82] is derived automatically from the OWL representation. The top level structure of the kitting workstation knowledge base is shown in Figure 16.

One requirement of the automated kitting application is that the data objects and solid objects with which the workstation deals must be represented. Data objects include designs of parts, designs of kits, locations, etc.—anything not made of matter. Solid objects are made of matter and include items such as the robot, actual parts, and actual kits. All solid objects have a location. Non-solid matter plays no role in kitting and is not represented. The distinction between data objects and solid objects is similar to the distinction between Abstract and Object made in SUMO [73] and the recently approved IEEE Standard for Ontologies for Robotics and Automation [83].

Solid objects in the kitting workstation model are required to have a location that is referenced to another solid object. For convenience, the workstation is located relative to itself; no other solid object may be located with respect to itself. Locations may be qualitative relative locations, specifying only that one object is "in" or "on" another object. Alternatively, locations may be quantitative, with a mathematical description of the location of the coordinate system of one object in the coordinate system of another – or simultaneously qualitative and quantitative. Each solid object must have a native coordinate system. Solid objects must have one primary location and may have multiple secondary locations. This is convenient, for example, in a simulation system, where the primary location is usually relative to a containing or holding object (e.g., a part in a parts tray) and the secondary location is with respect to the kitting workstation for the benefit of a graphic display system.

Solid objects are of two types, those called SkuObjects that are instances of stock keeping units (SKUs) and those called NoSkuObjects that are not. As shown in Figure 17, most of the information about a SkuObject is kept in the SKU it references. This includes the shape of the object, which may be expressed either or both as an internal stereotyped parametric shape or as an external shape described in some known shape format. NoSkuObjects may also have internal and/or external shapes. Parts, KitTrays, PartsTrays, etc. are SkuObjects. The robot, worktable(s), grippers, gripper changing station, etc. are NoSkuObjects.

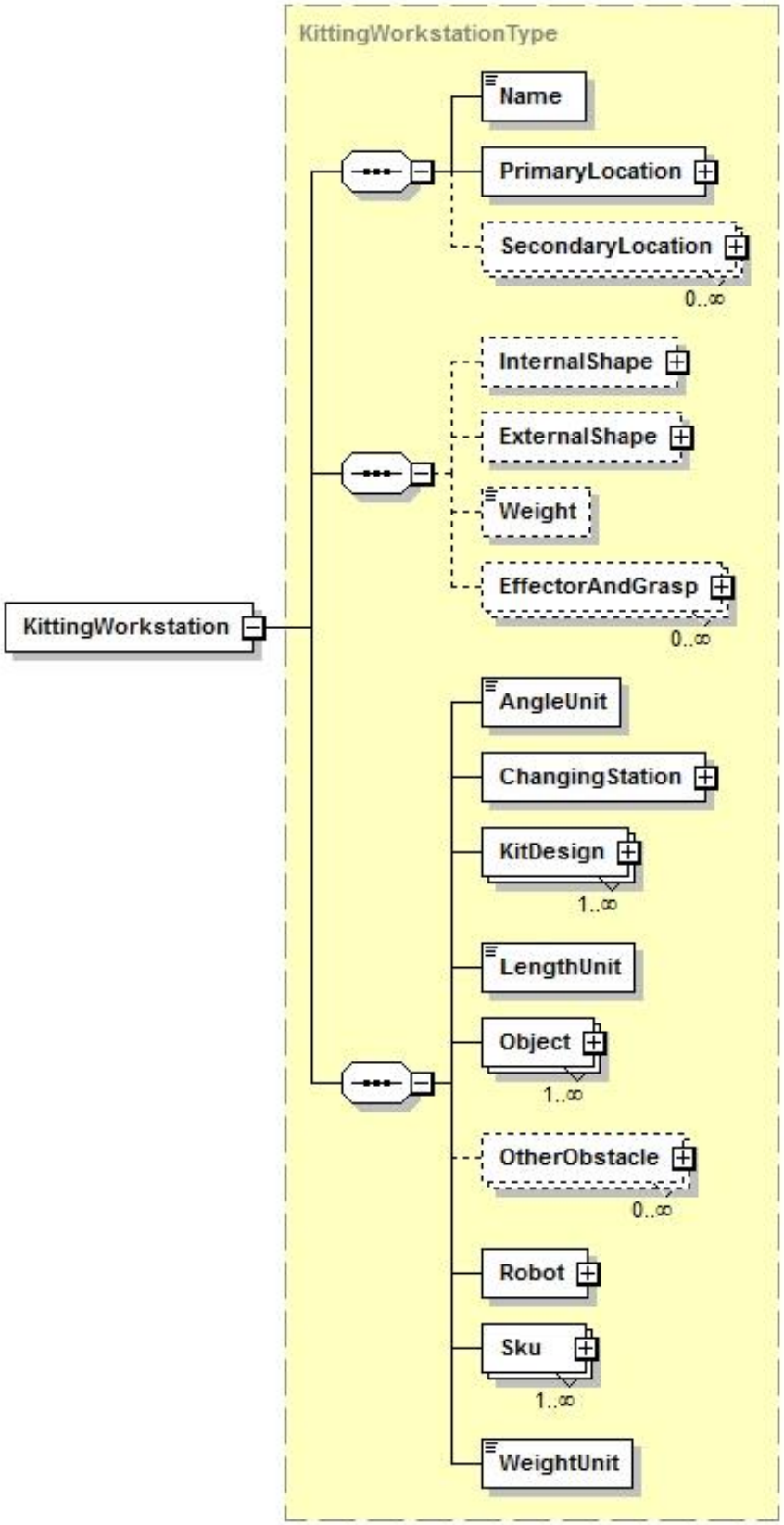

Figure 16: Kitting workstation knowledge base

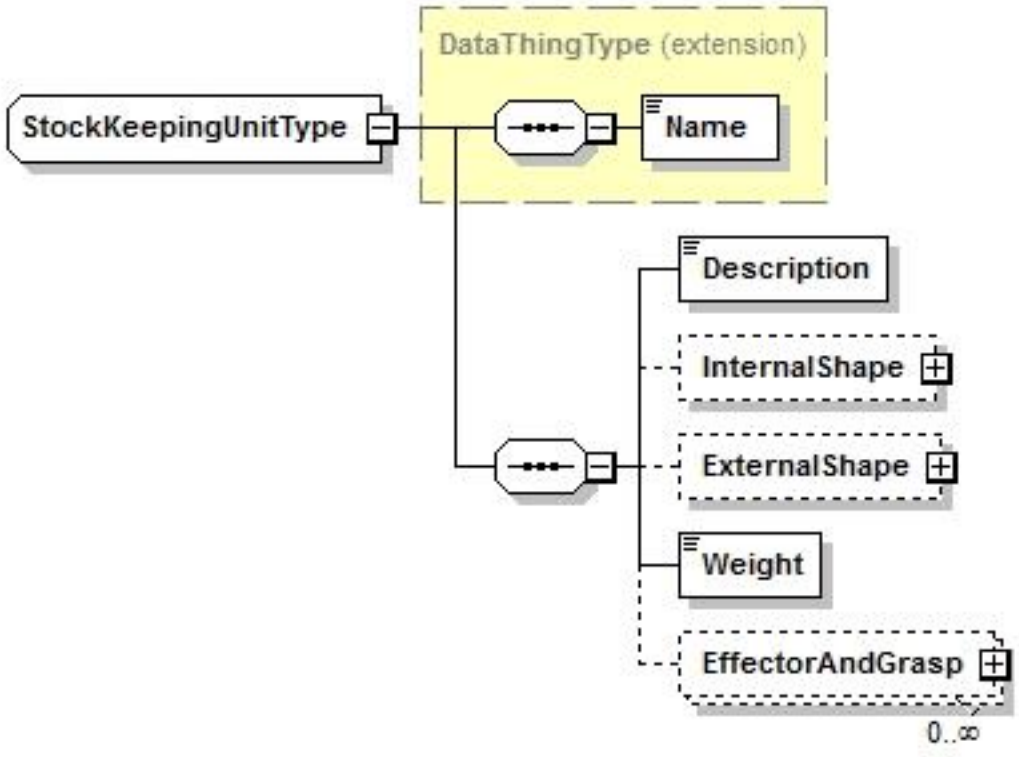

Figure 17: Stock keeping unit knowledge base

3. Discussion

While the taxonomy and performance metrics are focused on assembly tasks, many of the components are more generic, covering activities needed in many applications, such as placing, inserting, or aligning objects. Thus, developing a framework for evaluating assembly operations will have much wider benefits. This might provide an incentive to develop the performance metrics in an order that maximizes their utility across domains. On the other hand, some of the required activities, such as coordination and communication, will depend on the particular configuration of robots and associated equipment in the work cell, which would make the resulting performance evaluation specific to that single application.

When applying multiple performance measures, care needs to be taken to ensure that the results do not conflict. That is, measurement of components in isolation must be done in such a way that the results can be combined meaningfully, without "edge effect" in which the assumptions behind the individual performance measures do not match those for other components. For example, if one performance test results in an uncertainty in the position of the part that is greater than the tolerance of the subsequent step, then the global performance of the entire task is jeopardized. A real-world robotic application is likely to be complex and the interactions between task steps may not always be obvious. Rigorous statistical testing may be required to uncover some of the problems and to ensure that the evaluation is valid. In some sense, the task is similar to debugging code, in that all possible paths through the task should be evaluated.

One item that was not discussed in detail in the paper but is very important is how the combination of the performance of various tasks can be composed to predict the performance of a higher-level task. For example, if the performance of a "detect", "pick up", "align", and "screw" task can be determined with individual test methods, how can the performance of the overall "pick up and screw" operation (which is composed of a sequential combination of these four tasks) be determined? Much of this is a function of how dependent one task is on the previous tasks, or if the two tasks can be seen as independent. This will be the focus of future research.

Representation of the robot capabilities also poses some interesting research challenges. In the paper, we describe how various aspects of the representation have been handled by different efforts, but we have not found any research efforts that have tried to pull these aspects together into a single representation. On top of this, the compositionality issue described in the previous paragraph provides another aspect in which no relevant literature was found. In addition to representing the core concepts, we also need to ensure that the knowledge representation is usable by applications such as process planning systems so automated plans can be developed that leverage the capabilities of the robots and ensure the maximum probability that the plan executes properly.

4. Conclusions

We have proposed an initial set of performance metrics for robotic assembly, organized according to a taxonomy of assembly tasks. The performance considerations must include the range of environmental conditions. These include a geometric constraint perspective, the part properties, the difficulty of the assembly operations, and the starting and ending part positions and orientations. This organized way of presenting the different activities that comprise assembly, along with performance metrics serves as a way to guide and structure the development of a performance evaluation framework.[1] NIST has begun developing draft test methods for robotic assembly capabilities and expects to build on these efforts, based on the taxonomy initially defined in this document. Expanding the capabilities of robotic assembly holds great potential for gains in productivity and quality, not to mention reduction in ergonomic challenges. We have also explored the information requirements needed to represent these robot capabilities in a way that the knowledge can be directly used by manufacturing software systems. This assessment will serve as the basis for the development of such a knowledge representation. This document is intended to serve as a starting point for developing concepts and prototypes for test methods and artifacts that measure robotic system assembly capabilities with respect to the metrics described herein.

Bibliography

[1] International Federation of Robotics. (2012, Industrial Robot Statistics. *http://www.ifr.org/industrial-robots/statistics/*.

[1] See for example, http://www.nist.gov/el/isd/grasp.cfm for some preliminary grasping tests

[2] E. Messina, "Performance Standards for Urban Search & Rescue Robots: Enabling Deployment of New Tools for Responders," *Defense Standardization Program Office Journal*, pp. 43-48, 2007.
[3] B. Weiss and C. Schlenoff, "Evolution of the SCORE Framework to Enhance Field-Based Performance Evaluations of Emerging Technologies," Gaithersburg, MD.
[4] C. Schlenoff, M. Steves, B. Weiss, M. Shneier, and A. Virts, "Applying SCORE to Field-Based Performance Evaluations of Soldier Worn Sensor Technologies," *Journal of Field Robotics*, vol. 24, pp. 671-698, 2007.
[5] B. A. Weiss and C. Schlenoff, "Evolution of the SCORE framework to enhance field-based performance evaluations of emerging technologies," in *Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems*, 2008, pp. 1-8.
[6] C. Schlenoff, G. Sanders, B. Weiss, F. Proctor, M. P. Steves, and A. Virts, "Evaluating speech translation systems: Applying SCORE to TRANSTAC technologies," in *Proceedings of the 9th Workshop on Performance Metrics for Intelligent Systems*, 2009, pp. 223-230.
[7] G. Boothroyd, P. Dewhurst, and W. Knight, *Product design for manufacture and assembly. 1994*: Marcel Dekker, New York, 1998.
[8] J. Shi and R. Menassa, "Flexible robotic assembly in dynamic environments," presented at the Proceedings of the 10th Performance Metrics for Intelligent Systems Workshop, Baltimore, Maryland, 2010.
[9] D. E. Whitney, *Mechanical assemblies: their design, manufacture, and role in product development* vol. 1: Oxford university press, 2004.
[10] A. S. Kondoleon, *Cycle time analysis of robot assembly systems*: Society of Manufacturing Engineers, 1979.
[11] B. Hamner, S. Koterba, J. Shi, R. Simmons, and S. Singh, "An autonomous mobile manipulator for assembly tasks," *Autonomous Robots*, vol. 28, pp. 131-149, 2010.
[12] T. Suzuki, T. Ohashi, M. Asano, and S. Miyakawa, "Assembly reliability evaluation method (AREM)," in *Assembly and Task Planning, 2001, Proceedings of the IEEE International Symposium on*, 2001, pp. 294-299.
[13] S. Kmenta, B. Cheldelin, and K. Ishii, "Assembly FMEA: a simplified method for identifying assembly errors," in *ASME 2003 International Mechanical Engineering Congress and Exposition*, 2003, pp. 315-323.
[14] J. Sprovieri. (2004, May, 2004) Design for Robotic Assembly. *Assembly*. Available: http://www.assemblymag.com/articles/82786-design-for-robotic-assembly
[15] L. Seabra Lopes and L. M. Camarinha-Matos, "Towards intelligent execution supervision for flexible assembly systems," in *Systems, Man, and Cybernetics, 1996., IEEE International Conference on*, 1996, pp. 1225-1230 vol.2.
[16] X. Fiorentini, I. Gambino, V.-C. Liang, S. Rachuri, M. Mani, and C. Bock, "An Ontology for Assembly Representation, NIST Interagency/Internal Report (NISTIR) - 7436," Gaithersburg, MD 2007.
[17] M. Tenorth and M. Beetz, "KnowRob - Knowledge Processing for Autonomous Personal Robots," presented at the IEEE/RSJ International Conference on Intelligent Robots and Systems, St Louis, MO, 2009.
[18] D. Lenat, R. Guha, K. Pittman, D. Pratt, and M. Shephard, "CYC: Toward Programs with Common Sense," *Communications of the ACM*, vol. 33, pp. 30-49, 1990.

[19] T. Stipancic, B. Jerbic, and P. Curkovic, "Context-aware system applied in industrial assembly environment," *International Journal of Advanced Robotic Systems, Antonio Visioli (Ed.)*, vol. 9, 2012.
[20] S. Lemaignan, A. Siadat, J.-Y. Dantan, and A. Semenenko, "MASON: A proposal for an ontology of manufacturing domain," in *Distributed Intelligent Systems: Collective Intelligence and Its Applications, 2006. DIS 2006. IEEE Workshop on*, 2006, pp. 195-200.
[21] N. Lohse, "Towards an ontology framework for the integrated design of modular assembly systems," University of Nottingham, 2006.
[22] J. Huckaby and H. Christensen, "A Taxonomic Framework for Task Modeling and Knowledge Transfer in Manufacturing Robotics," presented at the Eighth International Cognitive Robotics Workshop, Toronto, Canada, 2012.
[23] J. Pfrommer, M. Schleipen, and J. Beyerer, "PPRS: Production skills and their relation to product, process, and resource," in *Emerging Technologies & Factory Automation (ETFA), 2013 IEEE 18th Conference on*, 2013, pp. 1-4.
[24] M. A. Goodrich, E. R. Boer, J. W. Crandall, R. W. Ricks, and M. L. Quigley, "Behavioral Entropy in Human-Robot Interaction," in *Proceedings of the 2004 Performance Metrics for Intelligent Systems (PerMIS) workshop*, Gaithersburg, MD.
[25] T. Tallinen, R. V. Osuna, J. L. M. Lastra, and R. Tuokko, "Product model representation concept for the purpose of assembly process modelling," in *Assembly and Task Planning, 2003. Proceedings of the IEEE International Symposium on*, 2003, pp. 222-227.
[26] B. P. Gerky and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *International Journal of Robotics Research*, vol. 23, pp. 939-954, 2004.
[27] H. A. Yanco and J. L. Drury, "A taxonomy for human-robot interaction," presented at the Proceedings of the AAAI Fall Symposium on HUman-Robot Interaction, AAAI Technical Report FS-02-03, 2002.
[28] A. Bloomfield, Y. Deng, J. Wampler, P. Rondot, D. Harth, M. McManus, *et al.*, "A taxonomy and comparison of haptic actions for disassembly tasks," presented at the Proceedings of the 2004 IEEE Virtual Reality Conference (VR'03), 2003.
[29] J. T. C. Tan, F. Duan, R. Kato, and T. Arai, "Collaboration Planning by Task Analysis in Human-Robot Collaborative Manufacturing System," in *Advances in RObot Manipulators*, E. Hall, Ed., ed: InTech, 2010.
[30] L. S. Homem de Mello and A. C. Sanderson, "Representations of mechanical assembly sequences," *Robotics and Automation, IEEE Transactions on*, vol. 7, pp. 211-227, 1991.
[31] K. W. Lyons, V. Rajan, and R. Sreerangam, "Representations and Methodologies for Assembly Modeling," National Institute of Standards and Technology NISTIR 6059, 1997.
[32] M. M. Baysal, U. Roy, R. Sudarsan, R. D. Sriram, and K. Lyons, "The open assembly model for the exchange of assembly and tolerance information: overview and example," in *ASME 2004 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 2004, pp. 759-770.
[33] S. Balakirsky, Z. Kootbally, C. Schlenoff, T. Kramer, and S. Gupta, "An industrial robotic knowledge representation for kit building applications," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, 2012, pp. 1365-1370.

[34] T. Kramer, Z. Kootbally, S. Balakirsky, C. Schlenoff, A. Pietromartire, and S. Gupta, "Performance evaluation of knowledge-based kitting via simulation," in *Automation Science and Engineering (CASE), 2013 IEEE International Conference on*, 2013, pp. 356-361.
[35] L. S. Haynes and G. H. Morris, "A formal approach to specifying assembly operations," *International Journal of Machine Tools and Manufacture,* vol. 28, pp. 281-298, 1988.
[36] G. Morris and L. S. Haynes, "Robotic assembly by constraints," in *Robotics and Automation. Proceedings. 1987 IEEE International Conference on*, 1987, pp. 1507-1515.
[37] J. D. Morrow and P. K. Khosla, "Sensorimotor primitives for robotic assembly skills," in *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on*, 1995, pp. 1894-1899.
[38] C.-H. Wu and M. G. Kim, "Modeling of part-mating strategies for automating assembly operations for robots," *Systems, Man and Cybernetics, IEEE Transactions on,* vol. 24, pp. 1065-1074, 1994.
[39] R. Konkar, "Incremental kinematic analysis and symbolic synthesis of mechanisms," Stanford University, 1993.
[40] International Organization for Standardization, "ISO 9283:1998 Manipulating industrial robots -- Performance criteria and related test methods.," ed, 1998.
[41] American National Standards Institute and Robotics Industries Association, "ANSI/RIA R15.05-2-1992 (R1999): Industrial Robots and Robot Systems - Path-Related and Dynamic Performance Characteristics - Evaluation," ed, 1992.
[42] American National Standards Institute and Robotics Industries Association, "ANSI/RIA R15.05-3-1992 (R1999): Industrial Robots and Robot Systems - Reliability Acceptance Testing - Guidelines," 1992.
[43] American National Standards Institute and Robotics Industries Association, "R15.05-1-1990 (R1999), Evaluation of Point-to-Point and Static Performance Characteristics of Industrial Robots and Robot Systems," ed, 1990.
[44] A. Jacoff, H.-M. Huang, E. Messina, A. Virts, and A. Downs, "Comprehensive standard test suites for the performance evaluation of mobile robots," in *Proceedings of the 10th Performance Metrics for Intelligent Systems Workshop*, 2010, pp. 161-168.
[45] N. D. M. Ceballos, J. A. Valencia, and N. L. Ospina, "Quantitative Performance Metrics for Mobile Robots Navigation," in *Mobile Robots Navigation*, A. Barrera, Ed., ed: InTech, 2010.
[46] L. Frommberger, T. Hildebrandt, and B. Scholz-Reiter, "User-Specified Performance Metrics for Autonomous Robots in Warehouse Logistics," in *Proceedings of the IROS 2011 Workshop on Metrics and Methodologies for Autonomous Robot Teams in Logistics*, ed, 2011.
[47] J. Marvel and J. Falco, "Best Practices and Performance Metrics Using Force Control for Robotic Assembly," NISTIR 7901, 2012.
[48] S. M. Singer and D. L. Akin, "A Survey of Quantitative Team Performance Metrics for Human-Robot Collaboration," presented at the 41st International Conference on Environmental Systems, Portland, Oregon, 2011.
[49] J. L. Burke, R. R. Murphy, D. R. Riddle, and T. Fincannon, "Task Performance Metrics in Human-Robot Interaction: Taking a Systems Approach," presented at the Workshop on Performance Metrics for Intelligent Systems, 2004.

[50] A. Steinfeld, T. Fong, D. Kaber, M. Lewis, J. Scholtz, A. Schultz, *et al.*, "Common metrics for human-robot interaction," presented at the Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction, Salt Lake City, Utah, USA, 2006.
[51] T. Fässberg, Å. Fasth, and J. Stahre, "A classification of carrier and content of information," presented at the CIRP International Conference on Assembly Technologies and Systems (CATS), Ann Arbor, MI, 2012.
[52] S. Patel and T. Sobh, "Manipulator Performance Measures - A Comprehensive Literature Survey," *Journal of Intelligent & Robotic Systems*, pp. 1-24, 2014/02/15 2014.
[53] J. Falco, J. Marvel, and E. Messina, "A Roadmap to Progress Measurement Science in Robot Dexterity and Manipulation," Gaithersburg, MD 2014.
[54] J. Falco, J. Marvel, and E. Messina, "Dexterous Manipulation for Manufacturing Applications Workshop," Gaithersburg, MD 2013.
[55] A. Ziout and A. Azab, "A cladistics approach to classification of joining and fastening methods," presented at the CIRP International Conference on Assembly Technologies and Systems (CATS) Ann Arbor, MI, 2012.
[56] "ASTM E2919-13: Standard Test Method for Evaluating the Performance of Systems that Measure Static, Six Degrees of Freedom (6DOF) Pose.," ASTM International 2014.
[57] S. Balakirsky, T. Kramer, Z. Kootbally, and A. Pietromartire, "Metrics and test methods for industrial kit building, National Institute of Standards and Technology, Gaithersburg, MD, USA, NISTIR 7942," 2013.
[58] B. R. Donald, "Planning multi-step error detection and recovery strategies," *The International Journal of Robotics Research*, vol. 9, pp. 3-60, 1990.
[59] A. Harris and J. M. Conrad, "Survey of popular robotics simulators, frameworks, and toolkits," in *Southeastcon, 2011 Proceedings of IEEE*, 2011, pp. 243-249.
[60] D. J. Miller and R. C. Lennox, "RIPE: A robot independent programming environment," in *Intelligent Robots and Computer Vision X: Algorithms and Techniques*, 1992, pp. 518-529.
[61] M. Clark, "JAUS Compliant Systems Offers Interoperability across Multiple and Diverse Robot Platforms," in *AUVSI Unmanned Systems North America Conference, Baltimore, Maryland*, 2005.
[62] *OpenJAUS*. Available: code.google.com/p/openjaus
[63] B. P. Gerkey, R. T. Vaughan, K. Stoy, A. Howard, G. Sukhatme, and M. J. Mataric, "Most valuable player: A robot device server for distributed control," in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, 2001, pp. 1226-1231.
[64] *The Player Project*. Available: playerstage.sourceforge.net
[65] H. Bruyninckx, "Open robot control software: the OROCOS project," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, 2001, pp. 2523-2528.
[66] *The OROCOS Project*. Available: www.orocos.org
[67] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, *et al.*, "ROS: an open-source Robot Operating System," in *ICRA workshop on open source software*, 2009, p. 5.
[68] *The Robot Operating System (ROS)*. Available: www.ros.org
[69] D. Edstrom, *MTConnect: To Measure Is To Know*: Virtual Photons Electrons, LLC Ashburn, VA.

[70] K. B. Lee, E. Y. Song, and P. S. Gu, "Integration of MTConnect and Standard-based Sensor Networks for Manufacturing Equipment Monitoring," in *ASME 2012 International Manufacturing Science and Engineering Conference collocated with the 40th North American Manufacturing Research Conference and in participation with the International Conference on Tribology Materials and Processing*, 2012, pp. 841-848.
[71] C. Schlenoff, E. Prestes, R. Madhavan, P. Goncalves, H. Li, S. Balakirsky, *et al.*, "An IEEE Standard Ontology for Robotics and Automation," presented at the International Conference on Intelligent Robots and Systems (IROS), Vilamoura, Algarve, 2012.
[72] ROS.org. (2015). *Unified Robot Description Model*. Available: wiki.ros.org/urdf/XML
[73] *Standard Upper Merged Ontology (SUMO)*. Available: http://www.ontologyportal.org/
[74] H. Mosemann and F. Wahl, "Automatic Decomposition of Planned Assembly Sequences Into Skill Primitives," *IEEE Transactions on Robotics and Automation*, vol. 17, 2001.
[75] J. Rosell, "Assembly and task planning using Petri nets: A Survey," *Journal of Engineering Manufacture*, vol. 218, 2004.
[76] "OWL-S 1.0 Release," The OWL Services Coalition, 2003.
[77] N. Pramanik, U. Roy, R. Sudarsan, R. Sriram, and K. Lyons, "Synthesis of geometric tolerances of a gearbox using a deviation-based tolerance synthesis scheme," in *ASME 2003 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 2003, pp. 209-216.
[78] W3C. (October 2004). *XML Schema Part 0: Primer Second Edition*. Available: www.w3.org/TR/xmlschema-0
[79] W3C. (October 2004). *XML Schema Part 1: Structures Second Edition*. Available: www.w3.org/TR/xmlschema-1
[80] W3C. (December 2012). *OWL 2 Web Ontology Language Primer (Second Edition)*. Available: www.w3.org/TR/owl2-primer
[81] W3C. (December 2012). *OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition)*. Available: www.w3.org/TR/owl2-syntax.
[82] MySQL. *MySQL Documentation: About MySQL Documentation*. Available: dev.mysql.com/doc/index-about.html
[83] IEEE, "Standard for Ontologies for Robotics and Automation (P1872-2015)," ed. Piscataway, NJ, 2015.